Generación de marcha de un robot humanoide imitando al ser humano

Victor Alfonzo Cornejo Arismendi

Orientador: Dr. Dennis Barrios Aranibar

Jurado:
Dr. Reinaldo A. C. Bianchi – Centro Universitário da FEI – Brasil
Dr. Pablo Javier Alsina – Universidade Federal do Rio Grande do Norte – Brasil
Dr. Juan C. Cutipa Luque – Universidad Nacional de San Agustin de Arequipa – Perú
Dr. José Eduardo Ochoa Luna – Universidad Católica San Pablo – Perú

Tesis presentada al
Departamento de Ciencia de la Computación
como parte de los requisitos para obtener el grado de
Maestro en Ciencia de la Computación.

Universidad Católica San Pablo – UCSP
Febrero de 2019 – Arequipa – Perú

A Dios, por todo lo que me ha dado, a Diana para que la use en futuras investigaciones y se desarrolle profesionalmente, a mis sobrinos, amigos y futuros hijos, si les interesa la inteligencia artificial o la robótica, esto es para ustedes.

Abreviaturas

CM Ciclo de Marcha

RH Robot Humanoide

RHs Robots Humanoides

FRI *Foot-Rotation Indicator*

GDL Grados de Libertad

MIMO *Multiple-input Multiple-output*

SLIP *Spring Loaded Inverted Pendulum*

HZD *Hybrid Zero Dynamics*

PCA *Principal Component Analysis*

EC Espacio de Configuración

AHRS *Attitude and Heading Reference System*

ZMP *Zero Moment Point*

3D Tres Dimensiones

RBF *Radial Basis Function*

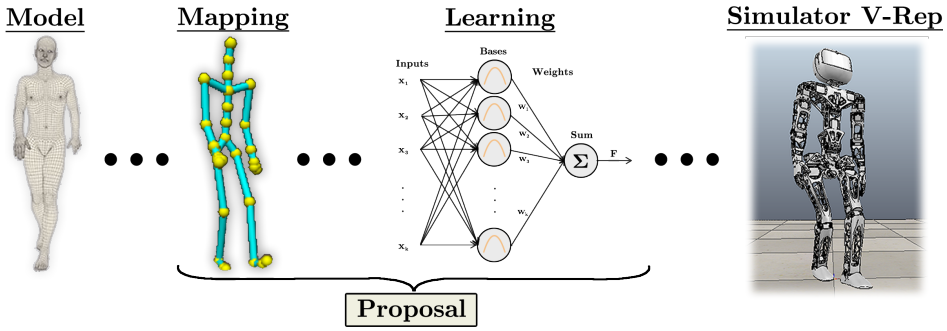
Agradecimientos

En primer lugar deseo agradecer a Dios por haberme guiado a lo largo de estos años de estudio. A mis padres Victor y Inés, sin su apoyo yo no sería como soy ahora. A mis hermanos Gustavo y Elisa, por sus buenos consejos y apoyo. A Diana Patiño Polar por ser la única persona que comprendió en su totalidad mis frustraciones y pesares en el transcurso del desarrollo de esta tesis, gracias por tu ayuda y apoyo incondicional.

Agradezco a mi orientador Prof. Dr. Dennis Barrios Aranibar, él fue mi guía a lo largo del desarrollo de esta tesis, gracias por perdonar mis errores y seguir a mi lado en los momentos difíciles. A mi Co-orientaador Dr. Luiz Marcos Garcia Gonçalves por haberme albergado en su hogar los días necesarios para poder acomodarme en Brasil, y por haberme brindado la oportunidad de hacer una pasantía en la UFRN. También agradezco a todos los miembros del laboratorio de robótica de la UCSP, por haber aguantado el calor de los computadores que ejecutaban mi código de tesis. Y a los amigos que me apoyaron, que con unas palabras de aliento me hicieron seguir adelante.

Deseo agradecer de manera especial al Consejo Nacional de Ciencia, Tecnología e Innovación Tecnológica (CONCYTEC) y al Fondo Nacional de Desarrollo Científico, Tecnológico e Innovación Tecnológica (FONDECYT-CIENCIACTIVA), que mediante Convenio de Gestión 234-2015-FONDECYT, han permitido la subvención y financiamiento de mis estudios de Maestría en Ciencia de la Computación en la Universidad Católica San Pablo (UCSP).

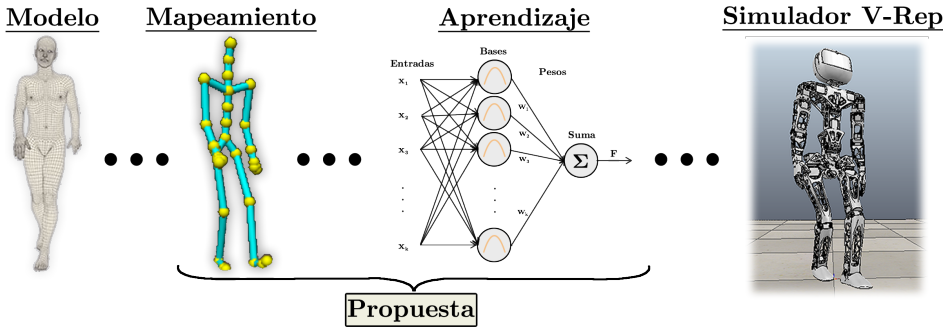
Abstract



This research poses the problem of generating a humanoid gait from the imitation of a human being, for this purpose, the kinematic information captured from human movement is used and applied in the proposal based on a first stage in the mapping from the points of capture of human movement to the articulations of humanoid robots, and in a second stage of learning from the generation of small disturbances of the mapped movements to the training of the learning model; In order to adjust the positions of the humanoid's joints to imitate human movement. The research shows the state of the art of related works and proposes a classification, differentiating between techniques with and without imitation. The proposal shows a scheme that takes the sequence of a human gait and replicates it in a humanoid robot using a mapping algorithm. These results are not sufficient, since the replication of movements does not solve the problem of equilibrium. For this reason, it is proposed to use a reinforcement learning algorithm that uses possible actions at each step and interpolates them in a neural network. This network uses a reward function that is given by the torso deviation angle that measures the stability of the robot. Convergence graphs of the proposal are also shown with different results using different test parameters to find the indicated convergence parameters. A similarity measure between the result of the proposal and the real human movement is also proposed. Concluding in an analysis of satisfactory results and proposals for future applications.

Keywords: Gait of Humanoids, Humanoid Robot, Human motion mapping, Learning by Imitation, Reinforcement Learning, Neural Network RBF, Sarsa(λ), RBF Sarsa(λ).

Resumen



Esta investigación plantea el problema de generar una marcha para un robot humanoide a partir de la imitación de un ser humano, para ello, la información cinemática capturada del movimiento humano es utilizada y aplicada en la propuesta basada en una primera etapa en el mapeo desde los puntos de captura del movimiento humano hasta las articulaciones de robots humanoides, y en una segunda etapa de aprendizaje desde la generación de pequeñas perturbaciones de los movimientos mapeados hasta el entrenamiento del modelo de aprendizaje, con el fin de ajustar las posiciones de las articulaciones del humanoide para imitar el movimiento humano. La investigación muestra el estado del arte de trabajos relacionados y plantea una clasificación diferenciando entre técnicas con y sin imitación. La propuesta muestra un esquema que toma la secuencia de una marcha humana y la replica en un robot humanoide usando un algoritmo de mapeamiento. Estos resultados no son suficientes, ya que la réplica de movimientos no resuelve el problema de equilibrio. Por ello se plantea utilizar algoritmo de aprendizaje por refuerzo que utiliza posibles acciones a cada paso y las interpola en esta red neuronal. Esta red utiliza una función de recompensa que está dada por el ángulo de desviación del torso que mide la estabilidad del robot. Así también se muestran gráficas de convergencia de la propuesta con distintos resultados usando diferentes parámetros de prueba para encontrar los parámetros indicados de convergencia. También se propone una medida de similitud entre el resultado de la propuesta y el movimiento real humano. Concluyendo en un análisis de resultados satisfactorios y propuestas de futuras aplicaciones.

Palabras clave: Marcha de Humanoides, Robot Humanoide, Mapeo de movimiento humano, Aprendizaje por Imitación, Aprendizaje por refuerzo, Red Neuronal RBF, Sarsa(λ), RBF Sarsa(λ).

Índice general

Índice de cuadros

Índice de figuras

Capítulo 1

Introducción

La robótica desde sus inicios fue creada con la finalidad de reemplazar al ser humano en tareas que tenían un alto costo de esfuerzo. Desde máquinas industriales hasta robots especializados, los robots sustituyeron trabajos pesados que el ser humano consideraba monótonos y repetitivos (Hockstein et al., 2007). En este sentido, existen tareas para las que es necesario que la capacidad de un robot sea lo más parecida a la del ser humano, es por ello el interés de desarrollar Robots Humanoides (RHs) que imiten la arquitectura del cuerpo humano.

Los RHs tienen una amplia aplicabilidad en la sustitución de trabajos que un hombre realice, ya sean para fines militares, de rescate, médico o espaciales. Los robots requieren de movimientos similares a los del ser humano para mejorar el desempeño de las tareas y automatizar procesos cotidianos para una persona. La investigación en RHs está motivada según la aplicabilidad que se le dé en cada una de las diferentes áreas que se investiga.

Una limitante latente en los RHs, es el problema de imitar los movimientos humanos que posean el mismo tamaño, torque y velocidad (Atkeson et al., 2018). Esto hace que el robot no pueda imitar en su totalidad al ser humano, ya que el hardware es limitante. Es decir, que para cada trabajo que se desee resolver, se requiere desarrollar e investigar un Robot Humanoide (RH) con las características de software y hardware necesarias que satisfagan dicha tarea. El software creado para estas tecnologías muestra un grado adaptativo según las limitaciones de la tecnología de la época (Endo et al., 2002).

1.1. Motivación y Contexto

Los robots humanoides tienen la finalidad de reemplazar las tareas de un ser humano, dado que el mundo esta hecho por humanos para humanos, es necesario crear un robot lo más parecido al ser humano para que pueda realizar las mismas tareas y

lograr el perfeccionamiento de cada una de ellas.

La marcha humana es una tarea vital de una persona y la principal que se utiliza para la locomoción. Al ser seres bípedos, la principal característica es el uso de piernas para el desplazamiento. Esta se compone básicamente en mover de forma coordinada cada una de las piernas de un lugar a otro para movilizar el torso manteniéndose erguido sin perder el equilibrio. El movimiento de las piernas se asemeja al desplazamiento de un péndulo (Poulakakis y Grizzle, 2009), donde cada pierna realiza un movimiento pendular de atrás hacia adelante de forma coordinada y una contraria a la otra.

La marcha o caminar es una tarea complementaria y fundamental para el funcionamiento del RH, habilita el movimiento de las piernas del robots para su traslación en el entorno (Fujiwara et al., 2003). Por ello, la marcha es un tema de estudio que sigue en perfeccionamiento de acuerdo a los avances tecnológicos de hardware y software.

La investigación de la marcha en humanoides es mayormente orientada a la estabilidad del robot y a la optimización en cuanto a tiempo y energía en un Ciclo de Marcha (CM). El hardware actual, si bien no iguala la capacidad humana, ha logrado un gran avance, y es por eso que RHs de menor tamaño y escala pueden lograr el movimiento de marcha de forma muy parecida a la del ser humano (Gouaillier et al., 2009).

Siguiendo con la motivación de Hirai et al. (1998), un RH debe parecerse al ser humano para que pueda realizar tareas domésticas de igual manera que el ser humano en su mismo entorno. Por ello es necesario desarrollar técnicas que utilicen los movimientos humanos como base para ser replicados en RHs, de tal manera que se asemejen al ser humano y al mismo tiempo demuestren una percepción más humana de la acción (Schaal, 1999).

Es así, que un RH con las mismas características del ser humano, podría reemplazarlo en tareas domésticas como servir un café o traer una bebida de la nevera y también en tareas de alto riesgo como reparar un satélite espacial o desactivar una bomba salvando vidas humanas en situaciones de peligro.

1.2. Planteamiento del Problema

En el transcurso de la evolución humana, el conocimiento del andar bípedo fue adquirido, aprendido, y utilizado desde nuestros ancestros, los RHs al igual que el ser humano necesitan de este conocimiento dado que su finalidad es realizar los mismos trabajos que realiza el ser humano. Por tanto, una vía para lograr esto es replicando los movimientos humanos en un RH. El principal problema de la imitación está en las diferencias estructurales y físicas que podrían existir entre el humano y el robot.

Otro problema que afecta a la investigación es la generalización de la marcha humana, La cadencia es el numero de pasos de un ser humano ejecutados en un periodo

de tiempo, esta varia según las características físicas de cada ser humano. Es difícil, conseguir el conocimiento genérico de una marcha humana es muy complejo dado que existen muchas variaciones de ellas alrededor del mundo.

Existen técnicas que resuelven el andar de un RH, pero estas técnicas comúnmente no imitan el andar humano porque la generación está basada en cálculos matemáticos y no utiliza la locomoción humana como base. En esencia, dado que ya existen investigaciones que han realizado con éxito el objetivo de hacer caminar al robot, entonces el problema que deseamos resolver en esta tesis es: ¿Cómo lograr hacer caminar un RH imitando al ser humano, generando una marcha lo más parecida a la humana?

1.3. Objetivos

El objetivo principal de la tesis es desarrollar técnicas y algoritmos que sirvan para generar una marcha de un robot humanoide imitando al ser humano. Esta tesis propone solucionar el problema propuesto con una técnica que sea capaz de generar un ciclo de marcha utilizando un andar humano y adaptarlo a un RH para que este mantenga el equilibrio.

1.3.1. Objetivos Específicos

Para cumplir este objetivo es necesario desarrollar los siguientes objetivos específicos:

- Recolectar información relevante de los trabajos relacionados de marcha de humanoides, clasificar y ordenar la información para encajar la propuesta en algún enfoque del actual estado del arte.
- Capturar, almacenar y refinar un CM humano. Obtener los datos de la marcha humana y visualizarlas en un simulador en Tres Dimensiones (3D).
- Crear una función de mapeo que transforme los datos cinemáticos de captura de un ser humano en Grados de Libertad (GDL) que pertenecen al modelo del RH en estudio.
- Desarrollar un algoritmo que adapte un CM humana para un RH sin que este pierda el equilibrio. Desarrollar un algoritmo que modifique la curva de la marcha de referencia y que la adapte a la marcha de un robot humanoide utilizando los datos cinemáticos de inclinación captados por los sensores del robot.
- Medir la similitud de un CM humana con la generada para un RH, para tener una noción de distancia y una magnitud para evaluar la propuesta.

1.4. Organización de la tesis

Esta tesis está compuesta por cinco capítulos, los cuales se desarrollan de la siguiente forma:

El primer capítulo da una introducción al tema de estudio desde una visión general a una especifica, planteando el problema y objetivos que se desarrollan a lo largo de la tesis.

El segundo capítulo muestra una recopilación de los trabajos relacionados de marcha de los RHs desde las primeras técnicas hasta la actualidad, proponiendo una clasificación de los distintos métodos y ubica la propuesta en el desarrollo actual.

El tercero muestra un marco teórico, que recopila todos los conocimientos previos necesarios para entender la propuesta, así como conceptos básicos y conceptos específicos, necesarios para el entendimiento del siguiente capítulo.

El cuarto capítulo presenta una propuesta basada en el estado del arte de imitación, donde se muestra un diagrama de flujo de cada parte y se explica en cada sección cada una de ellas de forma detallada. La propuesta consiste en mapear los movimientos humanos en un RH para luego generar perturbaciones que satisfagan todos los pasos de un CM; para esto es necesario aprender a utilizar la perturbación mas adecuada que se ajuste al humanoide, retro-alimentándose de los datos del equilibrio del robot. Todas las técnicas utilizadas en cada parte de la propuesta son mostradas en forma organizada conforme al diagrama inicial del capítulo.

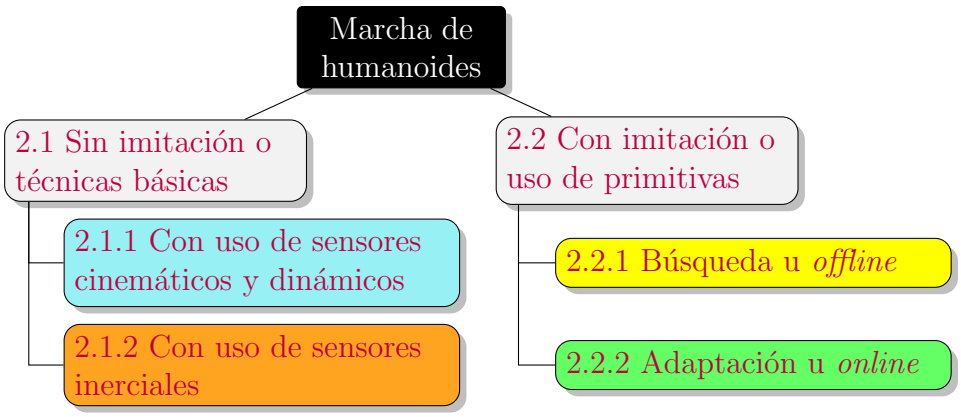
El quinto capítulo muestra los resultados del algoritmo propuesto de mapeamiento, mostrando resultados visuales y métricas para su debida interpretación. También se muestran resultados del método propuesto, así como distintas gráficas de resultados con distintos valores de prueba para encontrar una convergencia satisfactoria del algoritmo. Se muestra también una propuesta de métrica de similitud y se muestra resultados satisfactorios de la mejor solución encontrada. Para finalizar se muestra una análisis general de los resultados obtenidos

El sexto capítulo muestra las conclusiones y trabajos futuros, se muestra un análisis general de la investigación, se muestran conclusiones basados en los resultados hallados, además de las limitaciones que se tuvieron en el transcurso de la investigación, y recomendaciones para mejorar la investigación, así como también propuestas de trabajos futuros.

Capítulo 2

Trabajos Relacionados

En esta investigación presentamos el estado del arte de marcha de humanoides con una clasificación presentada en el Cuadro 2.1, la cual muestra una taxonomía basada en el uso o no de imitación con sus respectivas sub-clasificaciones explicadas a continuación.



Cuadro 2.1: Clasificación del estado del arte de marcha de humanoides

Los primeros métodos para solucionar el problema de marcha de humanoides se enfocaron en el equilibrio y optimización de movimiento con uso de sensores cinemáticos y dinámicos, es decir sensores de posición, fuerza y torque, que desarrollaron técnicas dependiendo del robot en uso, introduciendo conceptos básicos que se siguen utilizando hasta la actualidad. Una evolución a dichas técnicas se dieron gracias a la utilización de sensores inerciales como giroscopios y acelerómetros añadiendo nuevas técnicas que enriquecieron el uso de la retro-alimentación.

El otro enfoque para la solución de la marcha está dada por el uso de primitivas, que trata de imitar secuencias capturadas de movimientos y utilizarla en el robot. Este se divide en dos sub-clases de técnicas: técnicas de adaptación u *online* y de búsqueda u *offline*. Las técnicas de adaptación tratan de alterar las primitivas de movimientos para que el robot mantenga el equilibrio en tiempo real, es por ello que se llaman *online*. Las

técnicas de búsqueda u *offline*, convierten las primitivas en espacios reducidos lineales donde se aplican técnicas de búsqueda y optimización de soluciones para mantener el equilibrio del robot y al mismo tiempo se parezcan a la base primitiva, estas requieren de un pre-procesamiento, por ello se llaman *offline*.

En el Cuadro 2.2 se muestra el estado del arte de marcha de humanoides ordenada cronológicamente. La primera columna muestra al autor y la fecha ordenada de forma ascendente, la segunda muestra el tipo de actuador o motor que se utilizó en el robot de investigación, la tercera muestra los GDL del robot, la cuarta describe los sensores externos e internos que se utilizaron en la investigación, y la última columna describe el método utilizado. Además, las filas sombreadas de colores denotan un tipo de clasificación propuesta, donde:

Las filas de color celeste: Son las técnicas sin imitación que utilizan solamente sensores cinemáticos o dinámicos.

Las filas de color naranja: Son las técnicas sin imitación que utilizan los datos de sus sensores inerciales dentro de su propuesta.

Las filas de color amarillo: Son aquellas investigaciones que utilizaron algún tipo de primitiva humana y utilizan alguna técnica de búsqueda de una solución óptima.

Las filas de color verde: Son aquellas que utilizan algún tipo de técnica de imitación adaptativa en tiempo real u *online*.

Cuadro 2.2: Estado del arte de marcha de Robots Humanoides

Autor y fecha	Actuador	GDL	Sensores	Método
Zheng y Shen (1990)	Motor DC	8	Sensor de posición, sensor de fuerza	Esquema de control, métodos geométricos
Kajita et al. (1992)	Motor DC	4	Potenciómetro	Conservación de fuerza orbital con control proporcional derivativo
Kajita y Tani (1996)	Motor DC	6	Potenciómetros, *encoder* de pulso	Retroalimentación local
Shih (1996)	Servomotor DC	7	-	Conservación de *Zero Moment Point*
Goswami (1999)	Simulador	6	-	Control de *Foot-Rotation Indicator*
Pratt et al. (2001)	Motor Lineal	6	Sensor de fuerza y torque	Control híbrido de posición, fuerza e impedancia usando *Virtual Model* Control
Grizzle et al. (2001)	Motor DC	7	*Encoder* absoluto e incremental	Control dinámico con retro-alimentación local

Autor y fecha	Actuador	GDL	Sensores	Método
Westervelt et al. (2002)	Motor DC	7	*Encoder* absoluto e incremental	Control con *zero dynamics*
Chevallereau (2003)	Motor DC	7	*Encoder* absoluto e incremental	Limitaciones virtuales usando *zero dynamics*
Kajita et al. (2003)	Motor DC	7	*Encoder* absoluto e incremental	Control de trayectoria de los pies
Canudas-de Wit (2004)	Motor DC	7	*Encoder* absoluto e incremental	Limitaciones virtuales
Kim et al. (2006)	Simulador	25+	-	Control por imitación usando función de optimización
Lin et al. (2007)	Simulador	6	-	Control híbrido *Multiple-input Multiple-output*
Kwon et al. (2007)	Motor DC	35	Sensor de fuerza / torque, giroscopio	Control de impedancia
Hirose y Ogawa (2007)	Servomotor DC	12	Sensor de fuerza, giroscopio y acelerómetro	Control de predicción de movimiento
Chalodhorn et al. (2007)	Servomotor DC	25	Giroscopio	Optimización de movimiento basado en un modelo de aprendizaje predictivo usando primitivas
Ferreira et al. (2009)	Servomotor DC	6	Cámara y luces LED	Control de balance sagital usando captura de posición por cámara
Yang y Chong (2009)	Servomotor DC	25	Sensor de fuerza y presión	Control por imitación con aprendizaje de locomoción para mantener la dirección
Poulakakis y Grizzle (2009)	Simulado	6	-	Limitaciones virtuales usando *zero dynamics* usando retro-alimentación
Chevallereau et al. (2010)	Motor DC	7	*Encoder* absoluto e incremental	Control de retro-alimentación invariante en el tiempo
Li et al. (2010)	Servomotor DC	7	Sensor de fuerza	Control de estabilidad en los pies
Rokbani et al. (2010)	Servomotor DC	7	Sensor de Posición	Control usando optimización de enjambre de partículas

Autor y fecha	Actuador	GDL	Sensores	Método
Matsumoto y Kawamura (2010)	Servomotor DC	13	Sensor de fuerza / torque, giroscopio y acelerómetro	Control de dirección robusto
Kajita et al. (2010)	Servomotor DC	48	Sensor de fuerza, giroscopio, acelerómetro, posición, encoder incremental	Control de postura/ fuerza, usando Controlador Proporcional Integral Derivativo
Van Heerden y Kawamura (2010)	Servomotor DC	6	Sensor de fuerza, giroscopio, acelerómetro	Control basado en la observación del disturbio
Suwanratchatamanee et al. (2011)	Servomotor DC	17	Sensor táctil	Control distribuido usando háptica
Boutin et al. (2011)	Servomotor DC	25+	Sensor de fuerza / torque, giroscopio y acelerómetro	Control por imitación genérico de posición manteniendo el *Zero Moment Point*
Almetwally y Mallem (2013)	Servomotor DC	25	Kinect, sensor de fuerza / torque, giroscopio y acelerómetro	Control por imitación usando balance del centro de masa
Koenemann et al. (2014)	Servomotor DC	25	Xsens MVN, sensor de fuerza / torque, giroscopio y acelerómetro	Control por imitación usando balance del centro de masa
Hwang et al. (2014)	Simulador	25	-	Aprendizaje de un caminado estable usando Q-Learning
Chen et al. (2015)	Simulador	25+	-	Uso de primitivas de movimiento dinámico usando *autoencoders*
Teachasrisaksakul et al. (2015)	Servomotor DC	25	Sensor de fuerza / torque, giroscopio y acelerómetro	Control por imitación usando caminado dinámico
Lei et al. (2015)	Servomotor DC	25	Sensor de fuerza / torque, giroscopio y acelerómetro	Control por imitación con evaluación de similitud

Autor y fecha	Actuador	GDL	Sensores	Método
Hwang et al. (2016)	Servomotor DC	21	Sensor de posición	Control de imitación utilizando una solución a la cinemática inversa usando redes neuronales.
Lin y Nguyen (2016)	Servomotor DC	25	Kinect, sensor de fuerza / torque, giroscopio y acelerómetro	Control por imitación usando balance del centro de masa con planeamiento de trayectoria
Zhu et al. (2017)	Servomotor DC	25	Sensor de fuerza / torque, giroscopio y acelerómetro	Control de imitación usando Refinamiento Robusto Basado en Regresión
Kondo y Takahashi (2017)	Simulador	25	-	Control por imitación usando corrección con filtro de partículas y *autoencoders* jerárquico
Lin y Hwang (2017)	Servomotor DC	25	Sensor de fuerza / torque, giroscopio, acelerómetro y presión	Aprendizaje de balanceo de posturas usando *Q-Learning* con retro-alimentación de háptica en los pies
Kormushev et al. (2018)	Servomotor DC	12	Sensor de fuerza / torque, giroscopio, acelerómetro y presión	Aprendizaje por refuerzo aplicado a la trayectoria de la masa
Clever et al. (2018)	Servomotor DC	14	Sensor de fuerza / torque, giroscopio, acelerómetro y presión	Control de imitación usando modelos de cinemática inversa y generación de trayectorias
Khusainov et al. (2018)	Servomotor DC	22	Sensor de fuerza / torque, giroscopio, acelerómetro y presión	Planificación óptima de la trayectoria de la marcha del RH bípedo.
Hereid et al. (2018)	Servomotor DC	14	Sensor de fuerza / torque, giroscopio, acelerómetro y presión	Una formula escalable para la optimización de la marcha *Hybrid Zero Dynamics*

Cuadro 2.2: Estado del arte de marcha de Robots Humanoides ordenado cronológicamente.

2.1. Técnicas Básicas

2.1.1. Técnicas con uso de sensores cinemáticos y dinámicos

Desde un principio, en los 90's, la investigación de marcha de RHs se centró en el desarrollo de robots bípedos, robots que sólo tenían las extremidades inferiores y utilizaban sensores cinemáticos o dinámicos, las primeras investigaciones resolvían la cinemática inversa de las extremidades con métodos geométricos en superficies inclinadas (Zheng y Shen, 1990). El concepto de dinámica fue introducida usando la conservación de fuerza orbital (Kajita et al., 1992), que es un método para optimizar la fuerza en motores. Un avance relevante fue el uso de sensores *encoders* de pulso (Kajita y Tani, 1996) introduciendo la retro-alimentación sensorial.

El concepto *Passive Walking* (McGeer et al., 1990), fue introducido en los 90's, es una técnica importante que se basa en el movimiento pendular invertido, donde básicamente dos segmento que simboliza a las piernas se mueven de forma coordinada para mantener el equilibrio del robot y conservar el momento angular. El principal problema es su inestabilidad, pero fue una primera solución a la locomoción de RH.

El concepto *Zero Moment Point* (ZMP) en robots bípedos (Shih, 1996) (Vukobratović y Borovac, 2004) es un modelo matemático muy importante de anulación de fuerzas cinemáticas internas y externas para mantener el equilibrio del robot que fue utilizado mucho desde esa época hasta la actualidad. Esta técnica se basa en aplicar fuerzas de ajuste en torno a un pie de apoyo, esto hace una auto-corrección de la postura en tiempo real para mantener el equilibrio. Esta técnica marca un gran desarrollo en la investigación de robots bípedos, dado que se utiliza hasta la actualidad.

A inicios del siglo XXI se comenzó a tomar en consideración la diná0mica del robot al introducir muchos conceptos relevantes. Se desarrollaron modelos basados en el control del centro de masa, presentando el concepto de *Foot-Rotation Indicator* (FRI) (Goswami, 1999), el cual se centra en equilibrar la masa total del robot según el área de contacto de los pies de base, el concepto de *Virtual Model Control* (Pratt et al., 2001) el cual es muy importante en el desarrollo de los robot bípedos, utiliza sensores de tacto en las patas. Modelos dinámicos creados en base a modelos cinemáticos e impulso del centro de masa fueron alimentados por sensores para generar un modelo correctivo en tiempo real (Grizzle et al., 2001). El concepto de *Zero Dynamics* (Westervelt et al., 2002) es un modelo que anula las fuerzas dinámicas para lograr equilibrio mientras el robot se mueve.

A partir del 2003 el desarrollo de robots bípedos comerciales comenzó a surgir. El robot bípedo Rabbit (Grizzle et al., 2001), mostrado en la Figura 2.1a de 1.43 metros de altura y siete GDL sirvió como base para desarrollar la marcha tomando como estudio la trayectoria que deben llevar los pies (Chevallereau, 2003) (Kajita et al., 2003) e introducir el concepto de restricciones virtuales (Canudas-de Wit, 2004), que son restricciones generadas para cada articulación por la retro-alimentación de sensores.

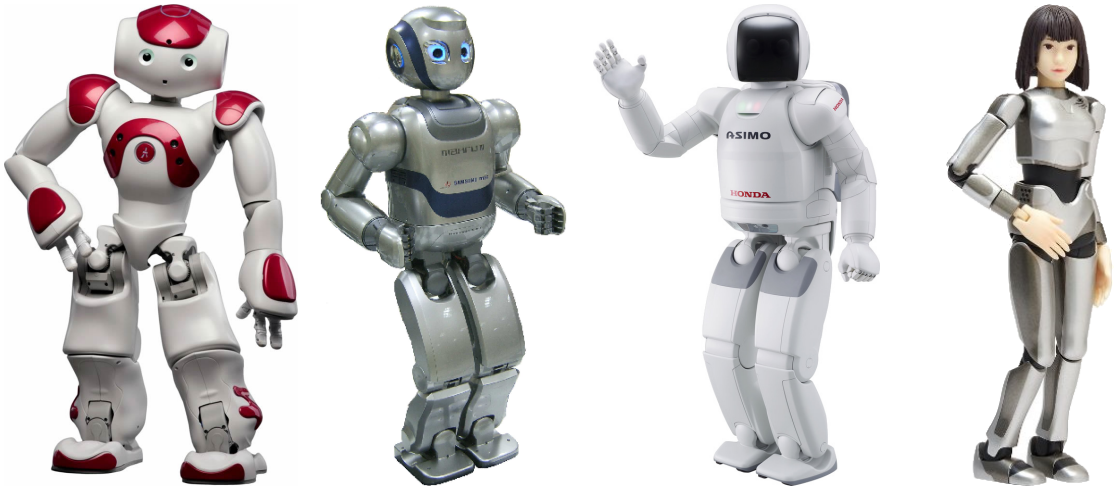
(a) Rabbit (Nguyen y Sreenath, 2015) (b) MABEL (Poulakakis y Grizzle, 2009)

Figura 2.1: Robots humanoides bípedos del siglo XXI

Otros enfoques utilizaron sistemas híbridos de control de múltiples entradas y múltiples salidas también llamados *Multiple-input Multiple-output* (MIMO) (Lin et al., 2007) que fueron desarrollados aplicando teoría de control utilizando una medida de incertidumbre para corregir el movimiento en tiempo de ejecución. Utilizando los datos de sensores se puede generar modelos dinámicos predictivos de centro masa y corregir la estabilidad en tiempo real para lograr equilibro en todo momento (Chevallereau et al., 2010).

El concepto de *Central Pattern Generators* (Ijspeert, 2008) consiste en la generación de patrones de movimientos para una correcta replicación, esta técnica se aplica no solo en bípedos, sino también a animales, el concepto se basa en generar patrones centrales de locomoción para poderlos replicar en otros modelos, para ello es necesario sensores inerciales y de movimiento para poder generar los patrones. Esta técnica es muy relevante en las investigaciones de locomoción de robots zoomórficos. En el 2009, con base en MABEL, el robot bípedo mostrado en la Figura 2.1b desarrolló el concepto *Spring Loaded Inverted Pendulum* (SLIP) (Poulakakis y Grizzle, 2009), consiste en mantener la fuerza inercial de una marcha en sólo dos puntos de contacto con el suelo y equilibrar la masa usando el concepto de *Hybrid Zero Dynamics* (HZD) (Poulakakis y Grizzle, 2009). Esta técnica importante se aplicó en un inicio en robots planares, es decir que su cálculo se basan en las dos dimensiones que pertenecen al plano sagital del robot.

El desarrollo más avanzado de sensores a partir del 2010, hizo que la utilización de sensores de tacto en los pies sea presentado como una solución básica para mantener el equilibrio de diferentes posturas (Li et al., 2010), así como el uso de sensores de presión distribuidos en la planta de los pies también llamado háptica (Suwanratchatamanee et al., 2011), que se utilizaron para generar modelos dinámicos centrados en la investigación de la posición del pie para mantener el equilibrio. Ya en técnicas más actuales (Khusainov et al., 2018), se propone regresar a métodos convencionales de planificación óptima de la trayectoria de la marcha, donde se presta atención en optimizar velocidad, aceleración y potencia de cada articulación, y otra solución utilizando una formula escalable para la optimización de la marcha HZD (Hereid et al., 2018), la cual



(a) NAO (b) Mahru III (c) ASIMO (d) HRP-4C

Figura 2.2: Robots humanoides comerciales con sensores inerciales

permite una generación rápida y confiable de caminar robótico dinámico de un caminar a través del marco HZD.

2.1.2. Técnicas con uso de sensores Inerciales

El sensor giroscopio es el encargado de medir la velocidad angular en los tres ejes cartesianos. La aparición del sensor giroscopio en la microelectrónica (Stokes, 1992) y su aplicación en los RHs (Batur et al., 2006) junto a los servomotores de alta precisión y fuerza dieron lugar al desarrollo de robots como ASIMO (Hirose y Ogawa, 2007) mostrado en Figura 2.2c, Mahru III (Kwon et al., 2007) mostrado en Figura 2.2b y NAO (Gouaillier et al., 2009) mostrado en Figura 2.2a, los cuales marcaron una etapa de desarrollo de hardware considerable.

Junto al desarrollo de nuevos robots en 2007, surgieron estudios de la dinámica en la marcha que tuvieron mejores resultados al tener una retroalimentación en tiempo real de la velocidad angular que actúa sobre el RH, como Mahru III (Kwon et al., 2007) y ASIMO (Hirose y Ogawa, 2007). Desarrollaron sistemas de control de impedancia para optimizar el uso de energía y de predicción de movimiento para poder corregir futuros errores, todo esto gracias al uso de sensores que miden la dirección, fuerza y torque.

El año 2010 presentó un avance significativo en la investigación, (Matsumoto y Kawamura, 2010) porque se desarrolló un control de dirección robusto que genera trayectorias y las modifica para mantener el equilibrio según los sensores. Sistemas más complejos de control de estabilidad utilizaron tres capas de control: posición, fuerza y postura (Kajita et al., 2010), en el robot japonés HRP-4C mostrado en la Figura 2.2d. Algunas investigaciones también utilizaron la generación de patrones de marcha (Van Heerden y Kawamura, 2010) (Kajita et al., 2010), el cual genera caminos espaciales de los movimientos de los pies.

2.2. Técnicas que usan imitación

2.2.1. Técnicas de búsqueda u *offline*

Por el lado de la imitación de marcha *offline*, investigaciones de marcha humana utilizaron la reducción multi-dimensional con algoritmos *Principal Component Analysis* (PCA) para poder generar un Espacio de Configuración (EC) (Lozano-Perez, 1990) lineal y continuo, para desde allí corregir la marcha con la utilización de algoritmos de aprendizaje y retro-alimentación sensorial (Chalodhorn et al., 2007). La captura de movimiento a partir del uso de marcadores visuales en las articulaciones más importantes del cuerpo humano con el uso de una cámara, ayudaron a replicar movimientos del robot de la cintura hacia arriba (Do et al., 2008). Investigaciones básicas de captura de movimiento para la replicación utilizaron luces LED ubicadas en la parte sagital del cuerpo humano, para luego ser capturadas con una cámara (Ferreira et al., 2009). Otras técnicas de optimización basadas en partículas (Rokbani et al., 2010) sirven para generar una marcha evaluando la estabilidad del robot en cada iteración.

Técnicas basadas en aprendizaje por refuerzo *Q-Learning* (Hwang et al., 2014) toman como base los datos capturados de una marcha con el sensor kinect para generar un CM y luego ser modificado según datos de sensores inerciales que sirven como recompensa para calificar las acciones del algoritmo. Un problema relevante en el estudio de las adaptaciones en tiempo real es la representación multi-dimensional, para ello se realiza una reducción de dimensiones en espacios continuos y lineales, una propuesta es la utilización de *Autoencoders*, los cuales generan funciones de compresión y descompresión de dimensiones. Una mejora de ello es la utilización de *Deep-Autoencoders* que añade capas convolucionales haciendo que la pérdida de datos en el proceso sea mucho menor (Chen et al., 2015).

Una mejora del anterior método es su aplicación en tiempo real con el uso de un Filtro de Partículas, que utiliza el espacio codificado del *Autoencoder* para modificar la curva de una marcha y adaptarla en tiempo real para mantener la estabilidad del robot utilizando retro-alimentación sensorial (Kondo y Takahashi, 2017). Otra técnica junta el método FRI junto con *zero dynamics* en una función de recompensa de *Q-learning* que modifican como base datos capturados por un Kinect para aprender en tiempo de ejecución una marcha estable (Lin y Hwang, 2017). Una solución más reciente aplica aprendizaje por refuerzo a la trayectoria de masa (Kormushev et al., 2018), el cual desarrolla un generador de patrón de caminar bípedo basado en ZMP de altura variable.

2.2.2. Técnicas de adaptación u *online*

Por otro lado, las técnicas de imitación por adaptación u *online* inicialmente se enfocaron sólo en los movimientos de la parte superior de un humanoide, utilizando una función de optimización que logra replicar los movimientos según las limitaciones

del robot (Kim et al., 2006). Una técnica de imitación muestra un aprendizaje de locomoción, modificando la postura de las piernas sin perder los momentos adecuados de presión en los pies y la dirección (Yang y Chong, 2009). Un avance generalizado de la técnica de imitación, muestra una solución genérica aplicada en dos tipos de robots, manteniendo el ZMP utilizando captura de movimiento por cámara (Boutin et al., 2011).

El robot Fujitsu HOAP-2 (Kormushev et al., 2011) creado en el 2006, incentivó el desarrollo de la imitación humana gracias a sus 25 GDL y el desarrollo de técnicas de imitación con control de balance y perturbaciones exteriores. La tecnología de cámaras infrarrojas como el sensor Kinect y el Robot NAO marcaron tendencia en la investigación de imitación, ya que el bajo costo del sensor y su software robusto genera modelos de *MOCAP* humanos en tiempo real, al mismo tiempo el robot NAO con 22 GDL se convirtió en un robot comercial y atractivo para su investigación (Almetwally y Mallem, 2013) (López Recio et al., 2013). Una visión reduccionista presenta una técnica utilizando imitación con cinemática inversa para modelos de trayectorias un andar humano (Clever et al., 2018), la propuesta modifica las trayectorias de los movimientos utilizando técnicas cinemáticas realizando aproximaciones a modelos que satisfagan el equilibrio.

La creación de Sensores IMU, y su avance en el 2014 con la introducción del sensor Xsens MVN para la captura de movimiento con el uso de sensores inerciales que implementan algoritmos *Attitude and Heading Reference System* (AHRS) en diferentes segmentos del cuerpo, impulsaron la imitación de movimientos aplicados en el robot NAO, su principal aporte es el control de balance en tiempo real (Koenemann et al., 2014).

Una técnica de imitación utilizando caminado dinámico adapta la posición actual y la modifica manteniendo el ZMP, un aporte importante es que propone una métrica para medir similitud entre la propuesta y el caminar humano (Teachasrisaksakul et al., 2015). Una propuesta se enfoca en la solución a la imitación de sólo los GDL de cintura para abajo, que adapta la posición sin el uso de sensores inerciales y basándose en la posición (Lei et al., 2015). Investigaciones actuales en cuanto a imitación de posiciones muestran un menor tiempo de adaptación, haciendo que la imitación sea más fluida (Lin y Nguyen, 2016).

Por otro lado, la aplicación de una red neuronal (Hornik et al., 1989) para solucionar la cinemática inversa de todo el robot involucra una etapa de aprendizaje a partir de imágenes tomadas por una cámara a personas en diferentes posturas, para luego usarla en tiempo real para su replicación (Hwang et al., 2016). Una propuesta actual es el *Refinamiento Robusto Basado en Regresión* (Zhu et al., 2017), que utiliza los índices de similitud espacial y de suavizado para poder calcular una regresión y mantener el equilibrio en un movimiento futuro.

Esta tesis está ubicada en la clasificación como técnica de imitación, la premisa es encontrar una solución que replique el movimiento humano del caminado, pero que al mismo tiempo esté abierto a futuras investigaciones de imitación general de cualquier

movimiento y distintas arquitecturas de robots. La idea central de la propuesta utiliza distintas técnicas de imitación para mantener el equilibrio (Westervelt et al., 2002) (Boutin et al., 2011) (Teachasrisaksakul et al., 2015) y lograr alterar la señal inicial de los movimientos con el uso de sensores de posición e inerciales (Chalodhorn et al., 2007) (Kwon et al., 2007) (Kajita et al., 2010) para lograr un equilibrio con auto corrección. (Teachasrisaksakul et al., 2015).

Capítulo 3

Marco Teórico

Dado el enfoque por imitación escogido para esta propuesta, es necesario definir algunos conceptos que ayuden a entender cada una de sus partes. Por ello en este capítulo se explica de forma breve cada uno de los conceptos introductorios de marcha humana, así como conocimientos previos necesarios para entender la propuesta.

3.1. Teoría de Marcha de Humanoides

En esta sección se explica la marcha humana, como funciona y cuales son sus fases, así como también se explica la arquitectura de un robot humanoide y como se relaciona cada una de sus articulaciones con la arquitectura humana.

3.1.1. Marcha humana

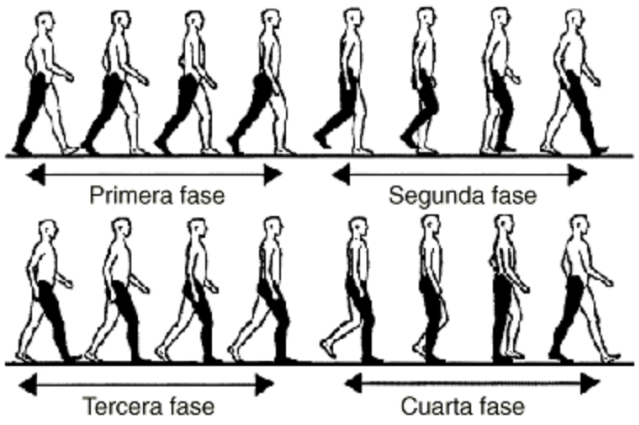


Figura 3.1: Diagrama de fases de un caminar humano (Nogueras et al., 1999)

.

La forma óptima de representación de una marcha es un CM (Harada et al.,

2006), el cual corresponde a una serie de movimientos dada por: Movimiento de la pierna derecha llevando el pie derecho de atrás hacia adelante y luego el pie izquierdo de la misma forma hasta llegar a la misma posición inicial.

El CM está dividido en fases que representan las etapas según como los pies están posicionados, la Figura 3.1 muestra las diferentes fases, la primera fase o *fase de despegue* es cuando el cuerpo que está apoyado en ambos pies tiene un impulso hacia adelante y busca mantener el peso en la pierna de adelante. La segunda fase o *fase de oscilación*, es cuando la pierna que queda en la parte trasera, es levantada y trasladada a la parte frontal, realizando un balanceo en la otra pierna de apoyo. La tercera fase o *fase de recepción de la carga*, se realiza desde el momento donde se recibe la carga del pie desplazado y se busca nuevamente, al igual que la primera fase, desplazar el peso del cuerpo a la pierna desplazada sin que ambos pies dejen de tocar el suelo. La cuarta fase o *fase media de apoyo* se realiza cuando la pierna posterior es levantada y desplazada a la parte frontal manteniendo un equilibrio constante y desplazando el peso del cuerpo usando un sólo pie de apoyo.

En resumen, todas estas fases componen el CM, el cual puede ser repetido cuantas veces se requiera, cabe resaltar que este análisis esta basado en un CM perfecto cuyo desplazamiento es hacia delante, modificaciones de este CM posibilitan el desplazamiento omnidireccional, los cuales no serán relevantes para esta investigación. Un punto importante a considerar, son los inicios y detención de marcha, en el caso de un inicio de marcha, se puede generar haciendo una modificación de la primera fase y reemplazarla por una posición rígida inicial donde los dos pies están posicionados uno al costado del otro, esto haría que la segunda fase se produzca a partir de esta posición y cualquiera de los pies realice un desplazamiento hacia adelante, impulsado el cuerpo hacia adelante. En el caso de una detención de marcha, es necesario realizar una modificación en la segunda o tercera fase, donde el pie a desplazarse debe ser posicionado al costado del otro y no delante del otro. Estas modificaciones son necesarios si se piensa en un inicio desde reposo y un final hacia el reposo.

3.1.2. Arquitectura de Humanoides

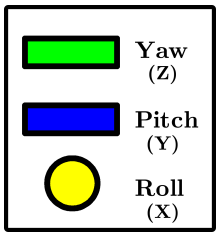
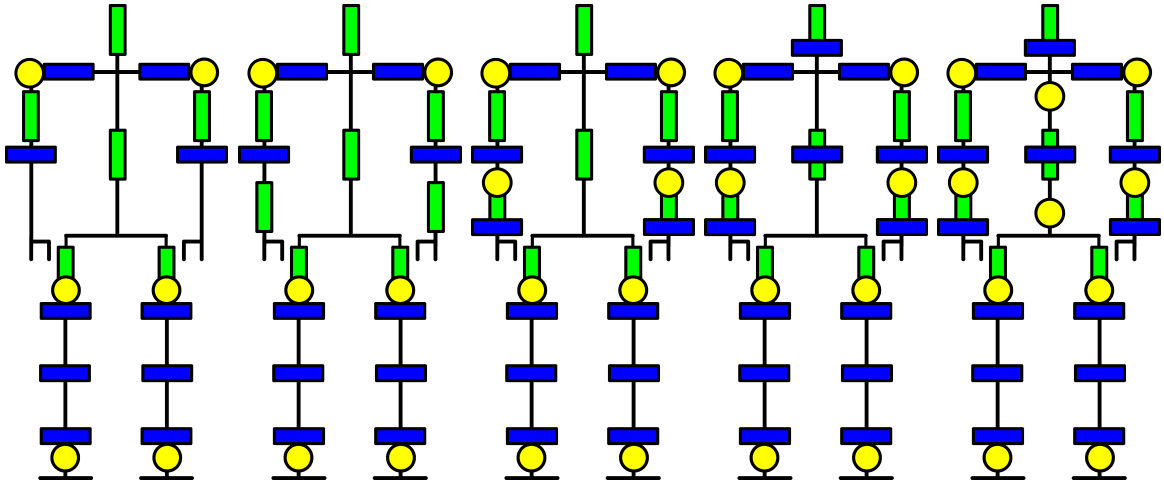


Figura 3.2: Leyenda de las articulaciones de la arquitectura propuesta

Para que estos movimientos, en este caso un CM, sean replicados se necesita de una arquitectura robótica capaz de imitar dichos movimientos para ejecutarlos de una forma similar a un ser humano. Todas las articulaciones tienen tres movimientos que simbolizan los *ángulos de euler* Baturone (2005), estas articulaciones serán simbolizadas



(a) 22 GDL (b) 24 GDL (c) 28 GDL (d) 30 GDL (e) 32 GDL

Figura 3.3: Ejemplos de arquitecturas de robots humanoides de 22,24,28,30,32 grados de libertad.

en la Figura 3.2 la cual muestra los tres tipos de articulaciones *Yaw* que simboliza el movimiento en el eje Z, *Pitch* que simboliza el movimiento en el eje Y, y *Roll* que es el movimiento en el eje X.

La Figura 3.3 muestra las diferentes arquitecturas de propuestas según la cantidad de GDL, La Figuras 3.3a y 3.3b muestran ejemplos de arquitecturas base de un humanoide, su principal diferencia entre ambas es el grado de libertad para cada una de las muñecas, esta articulación puede ser de cualquier tipo según el modelo del robot, La Figuras 3.3c y 3.3d se caracterizan por tener un juego de articulaciones completas en las manos, con la diferencia de una articulación adicional en el centro del torso y otro en el cuello. La Figura 3.3e tiene adicionalmente dos articulaciones transversales, las cuales generan un movimiento completo al torso.

El esquema de la Figura 3.3a representa un esquema base, el cual discrimina los movimiento secundarios y da prioridad a las articulaciones de las piernas. Tiene una única articulación en el Eje Z la cual permite el movimiento de cadera de forma parcial. Las demás arquitecturas adicionan articulaciones según la necesidad de su aplicabilidad, para esta investigación se prioriza el uso de las piernas, por ello el modelo de 22 GDL satisface la investigación.

Es necesario entender que la arquitectura de un RH es un parte importante, pero no es la única, también es importante tener en consideración la calidad de los motores a utilizar, se debe considerar la velocidad, aceleración y torque, los cuales en conjunto deben tener valores iguales a las extremidades humanas.

3.2. Conocimientos Previos

Para una buena comprensión de la propuesta de esta tesis es necesario conocer los siguientes conceptos:

3.2.1. Producto Cruz

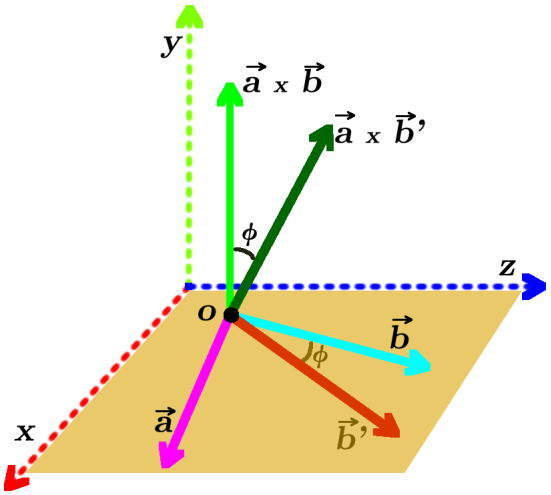


Figura 3.4: Esquema de funcionamiento del producto cruz en tres dimensiones

Para hallar la mayoría de vectores espaciales se utiliza el producto cruz, este tiene la capacidad de generar vectores normales del plano generados por dos vectores espaciales $\overrightarrow{a}$ y $\overrightarrow{b}$. La Figura 3.4 muestra una interpretación gráfica del funcionamiento del producto cruz en el espacio utilizado en la investigación, donde $\overrightarrow{a}$ y $\overrightarrow{b}$ son dos vectores pertenecientes al plano XZ, al tener esta naturaleza el producto cruz $a \times b$ será un vector paralelo al vector y, ahora, si tomamos un vector b' con una leve inclinación ϕ por debajo del plano XZ esta inclinación se reflejará en el vector $a \times b'$ el cual también se inclinará ϕ grados. Este comportamiento del producto cruz prueba su capacidad de generación de vectores normales al plano que contiene dichos vectores en el espacio.

3.2.2. Ángulo entre dos vectores en 3D

Es necesario definir una función que sirva para hallar los ángulos entre dos vectores, de un vector $\overrightarrow{a}$ a otro vector $\overrightarrow{b}$ en el espacio, para generar los ángulos de cada GDL que se utilizarán en el robot.

Definición **3.1** *angvec: Se denomina angvec a la función u operador que se encarga de medir el ángulo entre dos vectores en tres dimensiones y que está planteada en la ecuación 3.1.*

La ecuación 3.1, muestra una forma de hallar dicho ángulo, considerando el sentido de medición desde $\overrightarrow{a}$ hasta $\overrightarrow{b}$, sobre un plano cuya normal es $\overrightarrow{c}$,

$$angvec\left(\overrightarrow{a},\overrightarrow{b},\overrightarrow{c}\right) = sig\left(cos^{-1}\left(\frac{\bar{a'}.\bar{b}}{|\bar{a'}|.|\bar{b}|}\right)\right) \tag{3.1}$$

donde

$$\overrightarrow{a'} = \overrightarrow{c} \times \overrightarrow{a} \times \overrightarrow{c} \tag{3.2}$$

y

$$sig = Si \begin{cases} (\overrightarrow{a'} \times \overrightarrow{b}).\overrightarrow{c} < 0 & : -1 \\ otherwise & : 1 \end{cases} \tag{3.3}$$

La ecuación 3.1 sirve para hallar el ángulo entre dos vectores, donde:

- $\overrightarrow{a}$ es el vector referencia de donde se comienza a medir.
- $\overrightarrow{b}$ es el vector de llegada.
- $\overrightarrow{c}$ es el vector normal del plano de referencia para el cálculo del ángulo entre $\overrightarrow{a}$ y $\overrightarrow{b}$.

En el caso que el vector $\overrightarrow{a}$ no esté contenido en plano de la normal $\overrightarrow{c}$, la ecuación 3.2 se encarga de sacar la proyección a dicho plano, y la ecuación 3.3 se encarga de hallar el signo del ángulo, para saber si es positivo o negativo.

3.2.3. Red neuronal RBF

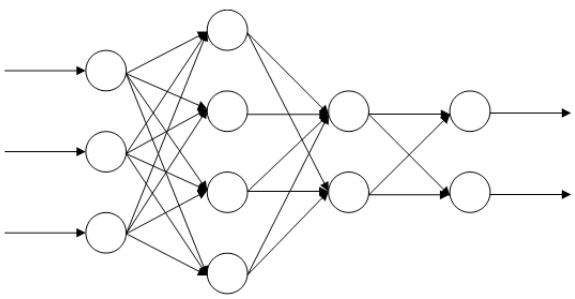


Figura 3.5: Esquema de Red Neuronal Multicapa Perceptron (Hornik et al., 1989)

Las redes neuronales son modelos que pueden servir como interpoladores en modelos no lineales y continuos, las cuales son una parte importante de esta propuesta. Una red neuronal es un modelo computacional que muestra una estructura jerárquica basada en grafos que trata de imitar el cerebro humano, cada nodo simboliza un neurona y cada conexión entre ellas simboliza una sinapsis. La Figura 3.5 muestra un ejemplo de una red neuronal multicapa, la cual se caracteriza por tener un conjunto de neuronas de entrada, un conjunto de neuronas ocultas, y un conjunto de neuronas de salida. Las neuronas de entrada son aquellas que se encargan de recepcionar los datos, y las de salida de expulsar la data, cada conexión sináptica tiene un peso w_{ij} que es un número cambiante que fluctuará en la etapa de entrenamiento de la red. Las neuronas de la capa oculta poseen un función de activación, esta característica hace que la curva generada por los pesos de la red neuronal se adapte según los datos de entrenamiento y genere un comportamiento matemático particular.

La funcionalidad de las redes neuronales tiene dos etapas de funcionamiento, la etapa de entrenamiento y la etapa de uso. En la etapa de entrenamiento, los pesos sinápticos y los valores de las funciones de activación comienzan de forma aleatoria, para luego cambiar y adaptarse utilizando algoritmos de *Back-propagation* (Hecht-Nielsen, 1992). Este algoritmo utiliza el error cuadrático de la salida, y retro-propaga el error desde la salida hasta la entrada actualizando los pesos sinápticos y valores de las funciones de activación.

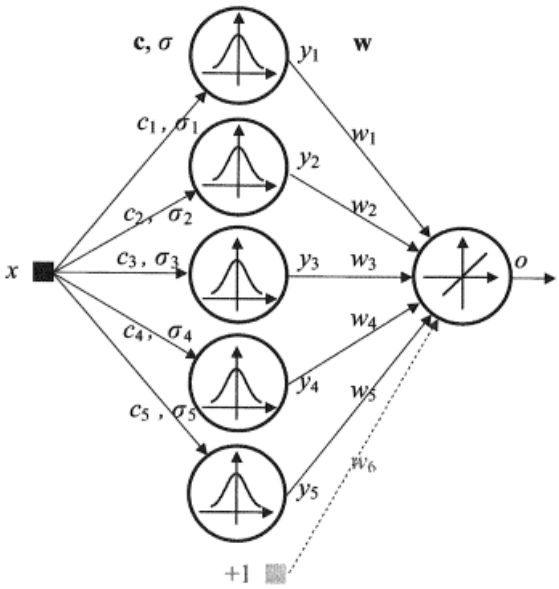


Figura 3.6: Esquema de Red Neuronal RBF (Chen et al., 1991)

Las redes neuronales tienen varias variantes, una de ellas es la red *Radial Basis Function* (RBF) (Chen et al., 1991) la cual tiene la caracteriza de poseer una función de activación basada en la función de base radial, una de ellas puede ser la función gaussiana no normalizada mostrada en la Ecuación 3.4,

$$\phi = \exp(-\frac{1}{\sigma^2}||c - x||^2) \tag{3.4}$$

Inicializar distribuyendo cada centro c_k con valores arbitrarios dentro un rango, para todo $k = 1, 2, 3, ..., n$

Repetir para cada par de datos (x_i, y_i):

$$\sigma^2 = \frac{1}{n} \sum_{j=1}^{n} ||c_i - c_j||$$

$$\phi_i = \sum_{k=1}^{n} -exp(\frac{1}{2\sigma_k^2} ||x_k - c_{ik}||^2)$$

$$w = \sum_{k=1}^{n} \phi_i^{-1} y_k$$

Hasta que terminar todos los pares (x_i, y_i).

Cuadro 3.1: Algoritmo de actualización de Red Neuronal RBF

donde:

- ϕ es la función de activación.
- σ^2 es la desviación estándar de la función en base radial.
- c son los centros de función en base Radial.
- x son las neuronas de entrada.

Esta función tiene la peculiaridad de poseer una arquitectura mostrada en la Figura 3.6 en la cual cada una de las neuronas ocultas tiene un valor σ^2 que es cambiado en la etapa de entrenamiento y un conjunto de centros c que son conexiones existentes para cada una de las neuronas de entrada. De la misma forma que una red multicapa, existen conexiones sinápticas de pesos w_{ij} que conectan las neuronas ocultas con las neuronas de salida y.

La fase de entrenamiento de una RBF tiene un método de adaptación donde los centros c se mueven y se adaptan según los datos de entrada x para generar una curva que interpola las diferentes funciones gaussianas. Los pesos w_{ij} también se actualizan en forma de matriz sacando una inversa de las salidas de las neuronas ocultas y multiplicándolas con la capa de salida, todo esto se muestra en el Cuadro 3.1 que muestra el algoritmo.

3.2.4. Sarsa (λ)

El algoritmo Sarsa (λ) (Gordon, 1996) es un algoritmo de aprendizaje por refuerzo que tiene un comportamiento discreto, el cual se utiliza en la propuesta con una

modificación para convertirlo en continuo.

Se caracteriza por tener el comportamiento de una máquina de estados, donde cada estado s_t simboliza los valores actuales de un sistema, y las conexiones entre estados son las acciones a_t, que son los valores variables entre un estado y otro. La transición de un estado a otro, por medio de una acción selecta, puede ser calificado por medio una recompensa r_t, cuyo valor es más alto si el camino es correcto, y será menor si el camino es erróneo.

Dado un estado inicial s_t se pueden generar un conjunto de posibles acciones $(a_{t_1}, a_{t_2}, a_{t_3}, ..., a_{t_n})$ que al ser ejecutadas llegan a un estado s_{t+1_n} o estado siguiente y el resultado puede ser evaluado con una función recompensa que da como resultado un valor r_{t+1}. El algoritmo Sarsa(λ) calcula el valor (s, a) estado/acción mostrado en la ecuación 3.6,

El cálculo del valor del estado/acción en Sarsa(λ) (Gordon, 1996) es realizado de la siguiente forma:

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha\delta_t e_t(s,a) \quad para\ todo\ s, a \tag{3.5}$$

donde:

$Q_t(s,a)$ es el estado/acción actual.

$Q_{t+1}(s,a)$ es el estado/acción futuro.

α es una constante de aprendizaje.

$e_t(s,a)$ es el valor de elegibilidad del par estado/acción.

y δ_t es el valor derivativo mostrado en la ecuación 3.6,

$$\delta_t = r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t) \tag{3.6}$$

donde:

r_{t+1} es la recompensa del estado siguiente ejecutado.

$Q_t(s_{t+1}, a_{t+1})$ es el estado/acción siguiente evaluado.

$Q_t(s_t, a_t)$ es el estado/acción actual evaluado.

γ es una constante de aprendizaje de δ_t.

y

$$e_t(s,a) = \begin{cases} \gamma\lambda e_{t-1}(s,a) + 1 & Si\ s = s_t\ y\ a = a_t \\ \gamma\lambda e_{t-1}(s,a) & Caso\ Contrario \end{cases} \quad para\ todo\ s,a \tag{3.7}$$

en caso de tramos acumulativos o

$$e_t(s,a) = \begin{cases} 1 & Si\ s = s_t\ y\ a = a_t \\ \gamma\lambda e_{t-1}(s,a) & Caso\ Contrario \end{cases} \quad para\ todo\ s,a \tag{3.8}$$

en caso de tramos con sustitución. Entonces se tiene que

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha(r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t))e_t(s,a) \tag{3.9}$$

donde:

$Q_t(s,a)$ es el valor de un par estado/acción cualquiera en tiempo t.

$Q_t(s_t,a_t)$ es un valor en tiempo t del par estado/acción ejecutado en tiempo t.

r_t es el retorno inmediato en el tiempo t.

α, λ y γ son parámetros del algoritmo.

$e_t(s,a)$ es el valor de elegibilidad del par estado/acción.

El conjunto de estados/acciones se guarda en una tabla discreta, donde las filas son los estados y las columnas son las acciones. Este algoritmo llena de forma recursiva toda esta tabla hasta llegar a un estado terminal s_{t_n}. Una característica de este algoritmo es el tratar con un conjunto finito de acciones iguales en cada estado, de la misma forma existe un conjunto finito de estados delimitado al inicializar el algoritmo. Para escoger el camino óptimo para llegar de un estado inicial a un estado final se utiliza una política derivada de Q la cual puede ser $greedy$ o $softmax$ para escoger las mejores acciones que lleven al estado terminal. Los tramos acumulativos toman como referencia conocimiento de iteraciones previas, mientras que los tramos con sustitución, simplemente comienzan en un valor inicial 1, descartando conocimiento previo en esa iteración. El algoritmo se muestra en el Cuadro 3.2.

Inicializar $Q(s,a)$ arbitrariamente y $e(s,a) = 0$ para todo s, a
Repite (para cada episodio)
  Inicializa s, a
  Repite
    Toma acción a y observa r, s'
    Selecciona a' de s' usando una política derivada de Q
    $\delta = r + \gamma Q(s', a') - Q(s, a)$
    $e(s, a) = e(s, a) + 1$
    Para todos s, a
      $Q(s, a) = Q(s, a) + \alpha\gamma e(s, a)$
      $e(s, a) = \gamma\alpha e(s, a)$
    $s = s'; a = a'$
  Hasta que s sea un estado terminal

Cuadro 3.2: Algoritmo de Aprendizaje Sarsa(λ)

3.3. Algoritmo de aprendizaje RBF-Sarsa(λ)

RBF-Sarsa(λ) es un algoritmo de aprendizaje por refuerzo con estados y acciones continuos, es una propuesta de Tesis de Maestria llamado: *Estratégias baseadas em aprendizado para coordenação de uma frota de robôs em tarefas cooperativas* (Barrios-Aranibar, 2005), el cual utiliza una función de aproximación y generalización con un método de aprendizaje.

El primer problema en la deducción del algoritmo RBF-Sarsa(λ) es determinar cuales son las entradas y salidas del mismo, siendo las entradas del algoritmo los estados y acciones del problema de aprendizaje por refuerzo, y la salida es definida como un valor de un par estado/acción $Q(s, a)$.

Ya que la salida del algoritmo será el valor del par estado/acción $Q(s, a)$, hay que denotar el valor deseado del algoritmo como el valor del par estado/acción del tiempo $t + 1$ $Q_{t+1}(s, a)$ que será calculado con base en el valor actual conocido por el algoritmo, esto quiere decir que sería calculado con base en la salida de la red en el tiempo actual. Así, se tiene que $Q_t(s, a)$ será obtenido ejecutando el algoritmo propuesto para un par estado/acción actual.

Utilizamos como base la ecuación 3.9, que es la ecuación que describe el algoritmo Sarsa(λ) y la reescribimos, se tiene que:

$$d_t(s,a) = d(t) = y_t(s,a) + \alpha(r_{t+1} + \gamma y_t(s_{t+1}, a_{t+1}) - y_t(s_t, a_t))e_t(s,a) \tag{3.10}$$

donde:

$d_t(s,a)$ es la salida deseada en la red neuronal RBF para el par estado/acción (s,a).

$y_t(s,a)$ es el valor de salida obtenido en la red para el par estado/acción$(s.a)$.

El calculo de salida en una red neuronal RBF está dado por:

$$y_t(s,a) = y(t) = \sum_{k=1}^{n} w_k(t)\phi_k(t) \tag{3.11}$$

donde:

$y_t(s,a)$ es el valor de salida obtenido en la red para el par estado/acción$(s.a)$.

$w_k(t)$ es un componente k del vector de pesos de la red neuronal en el tiempo t,

$\phi_k(t)$ es una función de base radial

n es un número de funciones de base radial de la red neuronal.

Considerando $\phi_k(t)$ como una función de base radial gaussiana, se tiene que:

$$y_t(s,a) = y(t) = \sum_{k=1} w_k(t) exp(\frac{1}{\sigma_k^2(t)}^{n} ||u(t) - c_k(t)||^2) \tag{3.12}$$

donde:

$y_t(s,a)$ es el valor de salida obtenido en la red para el par estado/acción$(s.a)$.

$\sigma_k^2(t)$ es una variación de la función gaussiana del neurona k.

$u(t)$ es un vector de entradas de la red neuronal.

$c_k(t)$ es un vector que representa el centro de la función de base radial de la neurona k.

Los parámetros de la red neuronal deben ser actualizados en cada iteración de la misma. Así se tiene que una actualización de los pesos usando una gradiente descendiente está dada por:

$$w_k(t+1) = w_k(t) - \eta_w \frac{\partial \varepsilon(t)}{\partial w_k(t)} \quad (para\ k = 1, 2, 3, ..., n) \tag{3.13}$$

donde n es el número de neuronas de la red neuronal RBF. La actualización de los centros de las funciones de base radial está dado por:

$$c_k(t+1) = c_k(t) - \eta_c \frac{\partial \varepsilon(t)}{\partial c_k(t)} \quad (para\ k = 1, 2, 3, ..., n) \tag{3.14}$$

es una actualización de las variancias de las funciones de la base radial, está dada por:

$$\sigma_k^2(t+1) = \sigma_k^2(t) - \eta_{\sigma^2} \frac{\partial \varepsilon(t)}{\partial \sigma_k^2(t)} \quad (para\ k = 1, 2, 3, ..., n) \tag{3.15}$$

donde

$$\varepsilon(t) = \frac{1}{2}(d(t) - y(t))^2 \tag{3.16}$$

entonces:

$$\varepsilon(t) = \frac{1}{2}[y_t(s,a) + \alpha(r_{t+1} + \gamma y_t(s_{t+1}, a_{t+1}) - y_t(s_t, a_t))e_t(s,a) - y_t(s,a)]^2 \tag{3.17}$$

y

$$\varepsilon(t) = \frac{1}{2}[\alpha(r_{t+1} + \gamma y_t(s_{t+1}, a_{t+1}) - y_t(s_t, a_t))e_t(s,a)]^2 \tag{3.18}$$

y finalmente

$$\varepsilon(t) = \frac{1}{2}\left[\alpha\left(r_{t+1} + \gamma \sum_{k=1}^{n} w_k(t)\phi_k^{(s_{t+1}, a_{t+1})}(t) - \sum_{k=1}^{n} w_k(t)\phi_k^{(s_t, a_t)}(t)\right) e_t(s,a)\right]^2 \tag{3.19}$$

donde $\phi_k^{(s_t, a_t)}(t)$ es una función de base radial del neurona k, validada en el par estado/acción (s_t, a_t) en el tiempo t.

Para el caso de la actualización de los pesos sinápticos de la red neuronal RBF se tiene que:

$$\begin{aligned} \frac{\partial \varepsilon(t)}{\partial w_k(t)} = &\left[\alpha\left(r_{t+1} + \gamma \sum_{k=1}^{n} w_k(t)\phi_k^{(s_{t+1}, a_{t+1})}(t) - \sum_{k=1}^{n} w_k(t)\phi_k^{(s_t, a_t)}(t)\right) e_t(s,a)\right] \\ &\left(\alpha\left(\gamma \phi_k^{(s_{t+1}, a_{t+1})}(t) - \phi_k^{(s_t, a_t)}(t)\right) e_t(s,a)\right) \end{aligned} \tag{3.20}$$

entonces:

$$\frac{\partial \varepsilon(t)}{\partial w_k(t)} = [d(t) - y(t)](\alpha(\gamma \phi_k^{(s_{t+1},a_{t+1})}(t) - \phi_k^{(s_t,a_t)}(t))e_t(s,a)) \tag{3.21}$$

$$\frac{\partial \varepsilon(t)}{\partial w_k(t)} = e(t)(\alpha(\gamma \phi_k^{(s_{t+1},a_{t+1})}(t) - \phi_k^{(s_t,a_t)}(t))e(s,a)) \tag{3.22}$$

Finalmente, se tiene que la ecuación para actualización de los pesos sinápticos de la red neuronal RBF con el algoritmo RBF-Sarsa(λ) está dada por:

$$w_k(t+1) = w_k(t) - \eta_w e(t)(\alpha(\gamma \phi_k^{(s_{t+1},a_{t+1})}(t) - \phi_k^{(s_t,a_t)}(t))e_t(s,a)) \tag{3.23}$$

Para el caso de la actualización de los centros de funciones de base radial de una red neuronal RBF, en el caso de la función gaussiana, se tiene que:

$$\frac{\partial \varepsilon(t)}{\partial c_k(t)} = \frac{\partial \varepsilon(t)}{\partial \phi_k^{(s_{t+1},a_{t+1})}(t)} \frac{\partial \phi_k^{(s_{t+1},a_{t+1})}(t)}{\partial c_k(t)} + \frac{\partial \varepsilon(t)}{\partial \phi_k^{(s_t,a_t)}(t)} \frac{\partial \phi_k^{(s_t,a_t)}(t)}{\partial c_k(t)} \tag{3.24}$$

$$\begin{aligned} \frac{\partial \varepsilon(t)}{\partial c_k(t)} = & \left[\alpha(r_{t+1} + \gamma \sum_{k=1}^{n} w_k(t) \phi_k^{(s_{t+1},a_{t+1})}(t) - \sum_{k=1}^{n} w_k(t) \phi_k^{(s_t,a_t)}(t))e_t(s,a)\right] \\ & \alpha\gamma w_k(t) e_t(s,a) \frac{2}{\sigma_k^2(t)} (u(t+1) - c_k(t)) \phi_k^{(s_{t+1},a_{t+1})}(t) - \\ & \left[\alpha(r_{t+1} + \gamma \sum_{k=1}^{n} w_k(t) \phi_k^{(s_{t+1},a_{t+1})}(t) - \sum_{k=1}^{n} w_k(t) \phi_k^{(s_t,a_t)}(t))e_t(s,a)\right] \\ & \alpha\gamma w_k(t) e_t(s,a) \frac{2}{\sigma_k^2(t)} (u(t) - c_k(t)) \phi_k^{(s_t,a_t)}(t) \end{aligned} \tag{3.25}$$

$$\begin{aligned} \frac{\partial \varepsilon(t)}{\partial c_k(t)} = & [d(t) - y(t)]\alpha\gamma w_k(t) e_t(s,a) \frac{2}{\sigma_k^2(t)} (u(t+1) - c_k(t)) \phi_k^{(s_{t+1},a_{t+1})}(t) - \\ & [d(t) - y(t)]\alpha\gamma w_k(t) e_t(s,a) \frac{2}{\sigma_k^2(t)} (u(t) - c_k(t)) \phi_k^{(s_t,a_t)}(t) \end{aligned} \tag{3.26}$$

$$\begin{aligned} \frac{\partial \varepsilon(t)}{\partial c_k(t)} = & \frac{[d(t) - y(t)]\alpha w_k(t) e_t(s,a) 2}{\sigma_k^2(t)} (\gamma(u(t+1) - \\ & c_k(t)) \phi_k^{(s_{t+1},a_{t+1})}(t) - (u(t) - c_k(t)) \phi_k^{(s_t,a_t)}(t)) \end{aligned} \tag{3.27}$$

$$\frac{\partial \varepsilon(t)}{\partial c_k(t)} = 2e(t)\alpha w_k(t)e(s,a)\frac{\Big(\gamma(u(t+1)-c_k(t))\phi_k^{(s_{t+1},a_{t+1})}(t)-(u(t)-c_k(t))\phi_k^{(s_t,a_t)}(t)\Big)}{\sigma_k^2(t)} \tag{3.28}$$

Finalmente, se tiene que una ecuación para actualización de los centros de las funciones de base radial de la red neuronal RBF con el algoritmo RBF-Sarsa(λ) está dada por:

$$\begin{aligned} c_k(t+1) =& c_k(t) - 2\eta_c e(t)\alpha w_k(t)e_t(s,a) \\ & \frac{\Big(\gamma(u(t+1)-c_k(t))\phi_k^{(s_{t+1},a_{t+1})}(t)-(u(t)-c_k(t))\phi_k^{(s_t,a_t)}(t)\Big)}{\sigma_k^2(t)} \end{aligned} \tag{3.29}$$

Para el caso de la actualización de las varianzas de las funciones de base radial de la red neuronal RBF, en el caso de las funciones gaussianas, se tiene que:

$$\frac{\partial \varepsilon(t)}{\partial \sigma_k^2(t)} = \frac{\partial \varepsilon(t)}{\partial \phi_k^{(s_{t+1},a_{t+1})}(t)}\frac{\partial \phi_k^{(s_{t+1},a_{t+1})}(t)}{\partial \sigma_k^2(t)} + \frac{\partial \varepsilon(t)}{\partial \phi_k^{(s_t,a_t)}(t)}\frac{\partial \phi_k^{(s_t,a_t)}(t)}{\partial \sigma_k^2(t)} \tag{3.30}$$

$$\begin{aligned} \frac{\partial \varepsilon(t)}{\partial \sigma_k^2(t)} =& \left[\alpha(r_{t+1}+\gamma\sum_{k=1}^{n} w_k(t)\phi_k^{(s_{t+1},a_{t+1})}(t) - \sum_{k=1}^{n} w_k(t)\phi_k^{(s_t,a_t)}(t))e_t(s,a)\right] \\ & \alpha\gamma w_k(t)e_t(s,a)\frac{||u(t+1)-c_k(t)||^2}{(\sigma_k^2(t))^2}\phi_k^{(s_{t+1},a_{t+1})}(t) - \\ & \left[\alpha(r_{t+1}+\gamma\sum_{k=1}^{n} w_k(t)\phi_k^{(s_{t+1},a_{t+1})}(t) - \sum_{k=1}^{n} w_k(t)\phi_k^{(s_t,a_t)}(t))e_t(s,a)\right] \\ & \alpha\gamma w_k(t)e_t(s,a)\frac{||u(t+1)-c_k(t)||^2}{(\sigma_k^2(t))^2}\phi_k^{(s_t,a_t)}(t) \end{aligned} \tag{3.31}$$

$$\begin{aligned} \frac{\partial \varepsilon(t)}{\partial \sigma_k^2(t)} =& [d(t)-y(t)]\alpha\gamma w_k(t)e_t(s,a)\frac{||u(t+1)-c_k(t)||^2}{(\sigma_k^2(t))^2}\phi_k^{(s_{t+1},a_{t+1})}(t) - \\ & [d(t)-y(t)]\alpha\gamma w_k(t)e_t(s,a)\frac{||u(t)-c_k(t)||^2}{(\sigma_k^2(t))^2}\phi_k^{(s_t,a_t)}(t) \end{aligned} \tag{3.32}$$

$$\begin{aligned} \frac{\partial \varepsilon(t)}{\partial \sigma_k^2(t)} =& [d(t)-y(t)]\alpha w_k(t)e_t(s,a) \\ & \frac{(\gamma||u(t+1)-c_k(t)||^2\phi_k^{(s_{t+1},a_{t+1})}(t) - ||u(t)-c_k(t)||^2\phi_k^{(s_t,a_t)}(t))}{(\sigma_k^2(t))^2} \end{aligned} \tag{3.33}$$

$$\frac{\partial \varepsilon(t)}{\partial \sigma_k^2(t)} = e(t) a w_k(t) e_t(s,a) \frac{(\gamma ||u(t+1) - c_k(t)||^2 \phi_k^{(s_{t+1}, a_{t+1})}(t) - ||u(t) - c_k(t)||^2 \phi_k^{(s_t, a_t)}(t))}{(\sigma_k^2(t))^2} \tag{3.34}$$

Finalmente, se tiene que la ecuación para actualización de las variables de las funciones de base radial de la red neuronal RBF como el algoritmo RBF-Sarsa(λ) es dada por:

$$\sigma_k^2(t+1) = \sigma_k^2(t) - \eta e_{\sigma^2} e(t) \alpha w_k(t) e_t(s,a) \frac{(\gamma ||u(t+1) - c_k(t)||^2 \phi_k^{(s_{t+1}, a_{t+1})}(t) - ||u(t) - c_k(t)||^2 \phi_k^{(s_t, a_t)}(t))}{(\sigma_k^2(t))^2} \tag{3.35}$$

Entonces, utilizando de forma coherente las ecuaciones 3.7, 3.8, 3.10, 3.23, 3.29 y 3.35 se obtiene el algoritmo de aprendizaje por refuerzo RBF-Sarsa(λ).

El algoritmo es mostrado en el Cuadro 3.3.

Inicializar los valores de los parámetros α,λ,γ,η_ω,η_c y η_{σ^2}
Inicializar w_k,c_k y σ_k^2con valores arbitrarios, para todo $k \longleftarrow 1, 2, 3, ..., n$
Inicializar el estado y la acción (s_0, a_0)
Repetir:
Ejecutar acción a_t, observar r_{t+1} y s_{t+1}
Escoger una acción a_{t+1} usando una política derivada de Q
Hacer u_t igual a el vector formado por s_t y a_t

$$y(s_t, a_t) \longleftarrow \sum_{k=1}^{n} w_k \phi_k^{(s_t, a_t)} \longleftarrow \sum_{k=1}^{n} w_k exp\left(\frac{1}{\sigma_k^2}||u_t - c_k||^2\right)$$

Hacer u_{t+1} igual a el vector formado por s_{t+1} y a_{t+1}

$$y(s_{t+1}, a_{t+1}) \longleftarrow \sum_{k=1}^{n} w_k \phi_k^{(s_{t+1}, a_{t+1})} \longleftarrow \sum_{k=1}^{n} w_k exp\left(\frac{1}{\sigma_k^2}||u_{t+1} - c_k||^2\right)$$

$$\delta_t \longleftarrow r_{t+1} + \gamma y(s_{t+1}, a_{t+1}) - y(s_t, a_t)$$

Hacer $e_t(s, a) \longleftarrow e_t(s, a) + 1$ si se quiere usar tramos acumulativos
o $e_t(s, a) \longleftarrow 1$ si se quiere usar sustitución de tramos
Para cada s, a visitado hasta el momento, hacer:
$e \longleftarrow \alpha\delta_t e_t(s, a)$
Para cada neurona k hacer:

$$w_k \longleftarrow w_k - \eta_w e\alpha \left(\gamma\phi_k^{(s_{t+1}, a_{t+1})} - \phi_k^{(s_t, a_t)}\right) e_t(s, a)$$

$$c_k \longleftarrow c_k - 2\eta_c e\alpha w_k e_t(s, a) \frac{\left(\gamma(u_{t+1} - c_k)\phi_k^{(s_{t+1}, a_{t+1})} - (u_t - c_k)\phi_k^{(s_t, a_t)}\right)}{\sigma_k^2}$$

$$\sigma_k^2 \longleftarrow \sigma_k^2 - \eta_{\sigma_k^2} e\alpha w_k e_t(s, a) \frac{\left(\gamma||u_{t+1} - c_k||^2\phi_k^{(s_{t+1}, a_{t+1})} - ||u_t - c_k||^2\phi_k^{(s_t, a_t)}\right)}{\left(\sigma_k^2\right)^2}$$

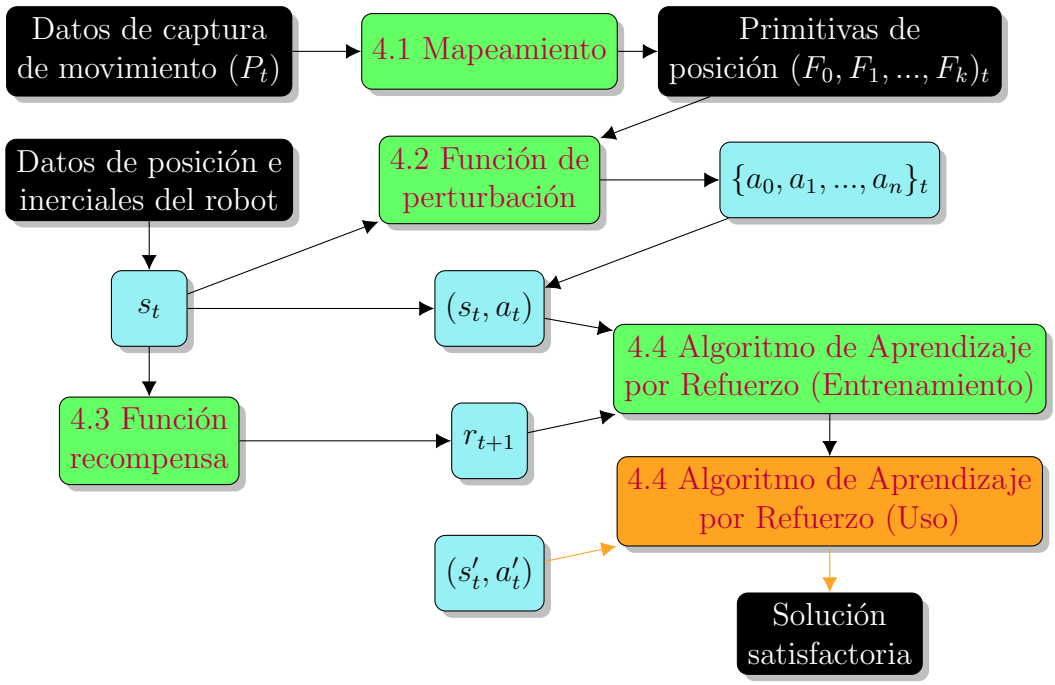
$e_t(s, a) \longleftarrow \gamma\lambda e_t(s, a)$
Hacer $s_t \longleftarrow s_{t+1}$ y $a_t \longleftarrow a_{t+1}$
Hasta que s_t sea un estado terminal

Cuadro 3.3: Algoritmo de Aprendizaje RBF-Sarsa(λ)

Capítulo 4

Propuesta

En el estado del arte, las técnicas de imitación presentan dos enfoques: adaptación y búsqueda. Las cuales se diferencian por la característica *online* u *offline*. Para la propuesta se eligió un método de búsqueda por imitación (*offline*), para obtener conocimiento de soluciones aceptables que mantengan el equilibrio del robot en un ciclo de marcha modificando levemente el andar original y así mantener la percepción original de un andar humano.



Cuadro 4.1: Diagrama de flujo de la propuesta

Definición **4.1** *Postura: Se denomina postura a la representación espacial de un fotograma de la escena de la simulación, es decir, es la unidad de presentación en cuanto al tiempo t de una simulación con n cantidad de posturas P_t*

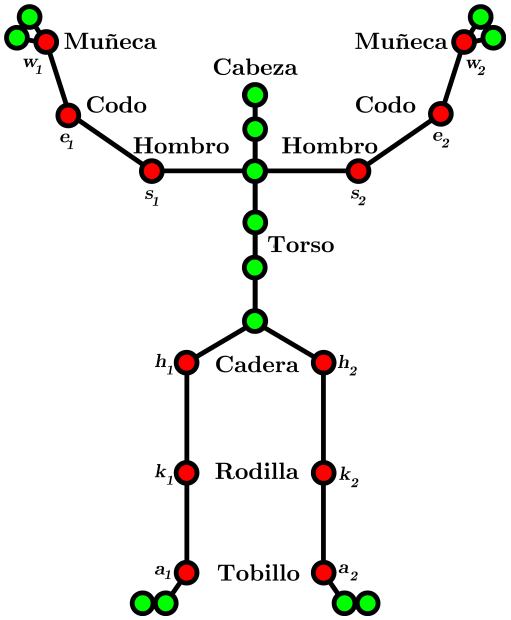


Figura 4.1: Diagrama de puntos básicos de un sistema de captura de movimiento de un ser humano, los puntos rojos son los puntos relevantes, y los verdes son los discriminados.

Definición **4.2** *Nube de Puntos: Una nube de puntos es un conjunto de puntos espaciales en 3D que simbolizan una postura de una escena de un CM, cada punto espacial simboliza una parte significativa de la posición de cada segmento del diagrama mostrado en la Figura 4.1.*

El cuadro 4.1 muestra una solución al problema propuesta, su enfoque consiste en la replicación de movimiento cinemático usando datos de captura de movimiento, para ello es necesario una base de datos que contenga CM de humanos. Esta base de datos se caracteriza por ser un conjuntos de posturas (Definición 4.1) que representan una posición en el tiempo t de una CM. Cada postura P_t contiene una nube de puntos (Definición 4.2) que servirá para su representación espacial en el simulador, la Figura 4.1 muestra como se representa cada dato y su correlación con el ser humano.

Para la propuesta se utiliza cualquier base de datos de captura de movimiento que contenga los puntos relevantes de la Figura 4.1, estos puntos P_t necesitan ser transformados a GDL, esta función se llama mapeamiento, la cual transforma cada postura a un conjunto de GDL, que llamaremos F_t. Para esta propuesta se desarrolla una solución genérica a un robot de de 22 GDL, de la cual se muestra su estructura en la Figura 4.2.

Con el mapeamiento se obtienen las posiciones de cada GDL del robot que se denominan F_t, las cuales sirven como referencia para generar acciones a_t utilizando una función de perturbación. La función de perturbación genera posibles acciones a partir de un F_t de forma aleatoria, para ello necesita del conjunto de posiciones actual de cada GDL del robot el cual será el estado s_t que servirá como referencia para generar posibles acciones futuras, esta función esta explicada en la Sección 4.2.

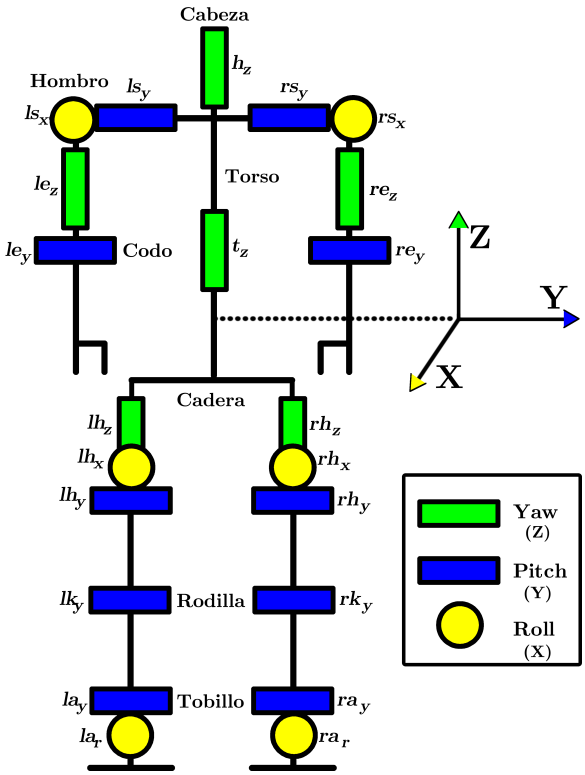


Figura 4.2: Arquitectura básica de Robot Humanoide con 22 grados de libertad que es utilizado en esta investigación (Omer et al., 2005).

Definición **4.3** *Función Recompensa: Es una función matemática que califica un estado s_t y le da un valor significativo que evalúa su desempeño.*

Se propone el uso de un algoritmo de aprendizaje por refuerzo para espacios de estados y acciones continuos, en este caso de estudio se propone usar el algoritmo RBF-Sarsa (λ) (Barrios-Aranibar, 2005). Esta técnica toma como base el Algoritmo de SARSA(λ) de la sub-sección 3.2.4, que a diferencia de la técnica de Q-learning, incorpora la política de control del agente y la añade en la actualización de valores, haciéndola más efectiva en el aprendizaje. El algoritmo SARSA(λ) tiene un comportamiento discreto, por lo que su autor propone juntarla con una Red Neuronal RBF para volverla continua, la razón por la que se usa esta red es por su peculiaridad al momento de detección de novedades, ya que tiene la característica de generar patrones nuevos al momento de la detección de datos de entrada inusuales. Ambas técnicas son de costo computacional bajo, que van acorde con el propósito de encontrar una propuesta eficiente y de rápida convergencia a diferencia de técnicas más complejas como *Deep Learning* (Perervenko, 2016).

Esta propuesta de aprendizaje por refuerzo, necesita de una función recompensa (Definición 4.3), esta utiliza los datos inerciales del estado actual del robot s_t y los evalúa para brindar una calificación r_{t+1} en cada iteración de las secuencias de posiciones. Con este proceso, la Red RBF-Sarsa (λ), realiza su proceso de aprendizaje hasta que los resultados sean satisfactorios, es decir que el robot realice un CM sin perder

el equilibrio, manteniendo el torso erguido en cada posición con un error mínimo. Con estos datos ya entrenados la red será capaz de calificar posibles acciones futuras (s'_t, a'_t) y escoger la mejor en el tiempo $t + 1$.

4.1. Mapeamiento

Para una transformación de puntos espaciales a ángulos se aplica cinemática inversa con métodos geométricos (Ferreira et al., 2009) basándose principalmente en hallar el ángulo entre dos segmentos dado un plano dependiente en tres dimensiones. Este plano comúnmente estará dado según la arquitectura del robot en uso, la Figura 4.2 muestra una arquitectura básica de robots humanoide (Kaneko et al., 2008), el cual presenta distintas articulaciones en los diferentes *ángulos de euler* (Baturone, 2005), por ello el algoritmo de mapeamiento depende directamente del robot que se utilice.

Básicamente, el algoritmo de mapeamiento toma como prioridad las restricciones físicas del robot y convierte los GDL del sistema de captura a posiciones correctas aplicables al modelo del robot. Por ello, no existe una función genérica, sino que se debe adaptar según la arquitectura del robot.

Dado el robot a utilizar en esta investigación, con la arquitectura de la Figura 4.2 de 22 GDL, el mapeo de cada extremidad será dependiente del plano de la unión del torso, por ello se explica primero el mapeo del torso, para luego explicar el mapeo de las extremidades superiores e inferiores.

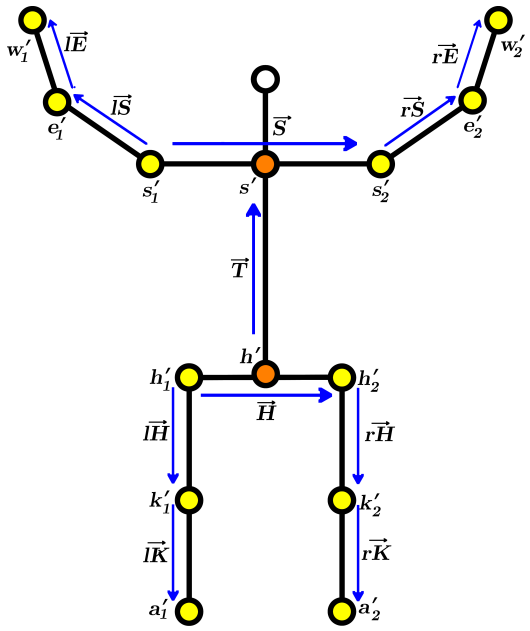


Figura 4.3: Modelo de mapeamiento del robot, mostrando puntos y vectores utilizados en las ecuaciones

4.1.1. Mapeamiento del Torso

Para el mapeo del torso se tomarán cuatro puntos significativos, s_1, s_2, h_1, h_2, de la Figura 4.1 respectivamente, donde:

(s_1, s_2) son los puntos pertenecientes a los hombros.

(h_1, h_2) son puntos pertenecientes a la cintura.

$\overrightarrow{S}$ son los vectores respectivos a los hombros, $\overrightarrow{S} = s_1 - s_2$.

$\overrightarrow{H}$ son los vectores respectivos a la cintura, $\overrightarrow{H} = h_1 - h_2$.

Un problema de estos vectores con respecto al robot, es que no están contenidos en el plano generado con la normal del Torso $\overrightarrow{T}$, mostrado en Figura 4.3, por ello es necesario generar las proyecciones de estos vectores para que sean contenidos en dicho plano.

Para hallar este eje rotacional t_z mostrado en la Figura 4.2 es necesario reducir y hallar nuevos puntos referenciales a los hombros y cintura que sean lo más parecidos a los originales.

$$s' = \frac{s_1 + s_2}{2} \tag{4.1}$$

$$h' = \frac{h_1 + h_2}{2} \tag{4.2}$$

Para ello se necesita hallar un punto s', que es el punto medio de referencia entre los dos hombros y está representado por la ecuación 4.1, también h' que es el punto medio de la cintura representado por la ecuación 4.2,

$$\overrightarrow{T} = h' - s' \tag{4.3}$$

con estos puntos podemos adquirir el vector dirección del tórax $\overrightarrow{T}$ en la ecuación 4.3.

Con ayuda del producto cruz se necesita hallar las normales de los planos generados por los vectores de los hombros y cintura.

$$\overrightarrow{NH} = \overrightarrow{T} \times \overrightarrow{H} \times \overrightarrow{T} \tag{4.4}$$

$$\overrightarrow{NS} = \overrightarrow{T} \times \overrightarrow{S} \times \overrightarrow{T} \tag{4.5}$$

Con estos vectores podemos hallar el vector $\overrightarrow{NH}$ y $\overrightarrow{NS}$ en las ecuaciones 4.4 y 4.5 respectivamente, que servirán como referencia para crear los puntos s'_1 y s'_2 pertenecientes a los hombros y h'_1 y h'_2 pertenecientes a la cintura.

$$s'_1 = s' + \frac{s' - \overrightarrow{NS}}{|s' - \overrightarrow{NS}|} \frac{|\overrightarrow{S}|}{2} \tag{4.6}$$

$$s'_2 = s' - \frac{s' - \overrightarrow{NS}}{|s' - \overrightarrow{NS}|} \frac{|\overrightarrow{S}|}{2} \tag{4.7}$$

$$h'_1 = h' + \frac{h' - \overrightarrow{NH}}{|h' - \overrightarrow{NH}|} \frac{|\overrightarrow{H}|}{2} \tag{4.8}$$

$$h'_2 = h' - \frac{h' - \overrightarrow{NH}}{|h' - \overrightarrow{NH}|} \frac{|\overrightarrow{H}|}{2} \tag{4.9}$$

Las ecuaciones 4.6, 4.7, 4.8, 4.8 muestran las soluciones de para los puntos referenciales s'_1, s'_2, h'_1 y h'_2 respectivamente. Estos puntos serán los nuevos puntos que se usarán como base para cada una de las extremidades. La Figura 4.3 muestra los puntos y vectores hallados en el modelo de mapeamiento para un mejor entendimiento. Con los vectores $\overrightarrow{NS}$ de la ecuación 4.5, $\overrightarrow{NH}$ de la ecuación 4.4 y $\overrightarrow{T}$ de la Figura 4.3 se puede encontrar el ángulo de rotación en el eje t_z usando la Definición 3.1 para encontrar el ángulo entre dos vectores de la siguiente forma:

$$t_z = angvec\,(\overrightarrow{NS}, \overrightarrow{NH}, \overrightarrow{T}) \tag{4.10}$$

4.1.2. Mapeamiento Extremidades Superiores

Se tienen nuevos puntos de base para cada extremidad, s'_1 y s'_2 son para las extremidades superiores. Es necesario obtener los puntos referenciales de cada extremidad en base a estos puntos, para ello se hallan los vectores $\overrightarrow{lS}$ y $\overrightarrow{lE}$ pertenecientes a los vectores dirección de brazo y antebrazo izquierdo y $\overrightarrow{rS}$ y $\overrightarrow{rE}$ pertenecientes a los vectores dirección de brazo y antebrazo derecho. Estos dos vectores son hallados a partir de los vectores del diagrama de puntos base de la Figura 4.1, usando los puntos de la siguiente forma:

$$\overrightarrow{lS} = s_1 - e_1 \tag{4.11}$$

$$\overrightarrow{lE} = e_1 - w_1 \tag{4.12}$$

$$\overrightarrow{rS} = s_2 - e_2 \tag{4.13}$$

$$\overrightarrow{rE} = e_2 - w_2 \tag{4.14}$$

para las siguientes ecuaciones se utilizará la función *angvec*, la cual se muestra en la Definición 3.1.

$$ls_y = -angvec\left(\overrightarrow{lS}, -\overrightarrow{T}, \overrightarrow{NS}\right) + 90° \tag{4.15}$$

$$ls_x = angvec\left(\overrightarrow{lS}, -\overrightarrow{T}, \overrightarrow{T} \times \overrightarrow{S}\right) + 90° \tag{4.16}$$

$$rs_y = angvec\left(\overrightarrow{rS}, -\overrightarrow{T}, \overrightarrow{NS}\right) - 90° \tag{4.17}$$

$$rs_x = angvec\left(\overrightarrow{rS}, -\overrightarrow{T}, \overrightarrow{T} \times \overrightarrow{S}\right) - 90° \tag{4.18}$$

Con estos vectores se hallan los ángulos ls_y, ls_x, rs_y, rs_x con las ecuaciones 4.15, 4.16, 4.17 y 4.18 respectivamente. Estas ecuaciones hallan los diferentes ángulos de pertenecientes a los hombros respecto a los planos dependientes del tórax en y y x.

$$le_y = -angvec\left(\overrightarrow{lE}, -\overrightarrow{lS}, \overrightarrow{lE} \times \overrightarrow{lS}\right) - 180° \tag{4.19}$$

$$re_y = angvec\left(\overrightarrow{rE}, -\overrightarrow{rS}, \overrightarrow{rE} \times \overrightarrow{rS}\right) + 180° \tag{4.20}$$

Para hallar los ángulos le_y y re_y se utilizan los vectores del brazo y del antebrazo para sacar el ángulo que existe entre ellos, las ecuaciones 4.19 y 4.20 muestran la solución.

Para los ángulos le_z y re_z es necesario crear vectores auxiliares que sirven como base para medir la variación de la posición actual a la posición si no existiese dicho ángulo.

$$\overrightarrow{lBt} = \overrightarrow{T} \times \overrightarrow{S} \times \overrightarrow{lS} \tag{4.21}$$

$$\overrightarrow{lBe} = \overrightarrow{lS} \times \overrightarrow{lE} \tag{4.22}$$

$$le_z = angvec\left(\overrightarrow{lBt}, \overrightarrow{lBe}, \overrightarrow{lBt} \times \overrightarrow{lBe}\right) \tag{4.23}$$

$$\overrightarrow{rBt} = \overrightarrow{T} \times \overrightarrow{S} \times \overrightarrow{rS} \tag{4.24}$$

$$\overrightarrow{rBe} = \overrightarrow{rS} \times \overrightarrow{rE} \tag{4.25}$$

$$re_z = -angvec\,(\overrightarrow{rBt}, \overrightarrow{rBe}, \overrightarrow{rBt} \times \overrightarrow{rBe}) \tag{4.26}$$

La ecuaciones 4.21, 4.22, 4.24 y 4.25 muestran los vectores auxiliares para cada antebrazo, y las ecuaciones 4.23 y 4.26 muestran los ángulos le_z y re_z hallados a partir de estos vectores auxiliares.

4.1.3. Mapeamiento Extremidades Inferiores

Todos los ángulos a continuación nombrados, se muestran en la Figura 4.2. Para hallar los ángulos de la cintura es importante hallar la normal ortogonal de la rodilla la cual se define:

$$\overrightarrow{lKn} = \overrightarrow{lH} \times \overrightarrow{lK} \tag{4.27}$$

$$\overrightarrow{rKn} = \overrightarrow{rH} \times \overrightarrow{rK} \tag{4.28}$$

$$lh_z = -90° - angvec\,(\overrightarrow{T} \times \overrightarrow{H}, \overrightarrow{lKn}, \overrightarrow{T}) \tag{4.29}$$

$$lh_y = 90° - angvec\,(\overrightarrow{lH}, \overrightarrow{T} \times \overrightarrow{H}, \overrightarrow{lKn}) \tag{4.30}$$

$$lh_x = angvec\,(\overrightarrow{lKn}, \overrightarrow{NH}, \overrightarrow{T} \times \overrightarrow{H}) \tag{4.31}$$

$$rh_z = 90° - angvec\,(\overrightarrow{T} \times \overrightarrow{H}, \overrightarrow{rKn}, \overrightarrow{T}) \tag{4.32}$$

$$rh_y = -90° + angvec\,(\overrightarrow{rH}, \overrightarrow{T} \times \overrightarrow{H}, -\overrightarrow{rKn}) \tag{4.33}$$

$$rh_x = angvec\,(\overrightarrow{rKn}, -\overrightarrow{NH}, \overrightarrow{T} \times \overrightarrow{H}) \tag{4.34}$$

$$lk_y = angvec\,(\overrightarrow{lH}, \overrightarrow{lK}, \overrightarrow{lKn}) \tag{4.35}$$

$$rk_y = angvec\,(\overrightarrow{rH}, \overrightarrow{rK}, -\overrightarrow{rKn}) \tag{4.36}$$

Para hallar las tres articulaciones de la cintura lh_z, lh_y, lh_x del lado izquierdo y rh_z, rh_y, rh_x del lado derecho es necesario definir las ecuaciones 4.29, 4.30, 4.31 y 4.32, 4.33, 4.34 respectivamente, en este caso es necesario hallar los ángulos respecto al espacio generado en el punto h_1 y el punto h_2 para que a partir de allí se midan los ángulos de los tres ejes. Para los ángulos lk_y y rk_y se definen en las ecuaciones 4.35 y 4.36, que son los ángulos formados por los vectores del fémur y del húmero de cada pierna.

Cabe destacar que los GDL de los pies serán discriminados en esta investigación y no mapeados, ya que las técnicas para hallarlos pueden ser variables según el robot, esto depende directamente de la cantidad de GDL que tengan los pies, se sugiere utilizar el enfoque de FRI (Goswami, 1999) que presenta un enfoque de balanceo según la superficie en contacto que existe entre el pie y el suelo. O el enfoque de punto de contacto ZMP (Tedrake et al., 2015), donde los pies sólo se consideran como si fueran zancos y se considera el modelo dinámico para mantener el equilibrio.

4.2. Función de perturbación

La función de perturbación sirve para generar conjunto de acciones a partir de una posición en un tiempo t, es decir, para una posición s_t se genera un grupo de acciones $(a_0, a_1, a_2, ..., a_n)_t$ donde n es la cantidad acciones que se desea utilizar.

Las acciones son posibles movimientos futuros que serán evaluados en el simulador y obtendrán una recompensa al ejecutarse. Este conjunto de datos generados servirá en cada generación de ejecución y necesita un algoritmo para su generación. Una acción es un conjunto de posiciones, en este caso de 22 GDL, es aconsejable no perturbar de forma homogénea todos los GDL, sino generar posibles soluciones dependiendo del rango de variabilidad de cada articulación, para ello se define un vector de constantes de variabilidad $(\epsilon_0, \epsilon_1, ..., \epsilon_n)_t^m$ que tiene la misma longitud que la cantidad de GDL y que se genera de forma aleatoria dentro del rango de variabilidad mencionado. La ecuación 4.37 representa la función de perturbación propuesta,

$$a_t^m = (F_{t+1} - s_t) + \epsilon_t^m \tag{4.37}$$

donde:

m es la cantidad total de acciones en el tiempo t.

s_t simboliza la posición actual del robot.

F_{t+1} es la postura mapeada futura a la que se quiere llegar.

ϵ_t^m es la perturbación generada m en la iteración t.

4.3. Función de recompensa

La función de recompensa utiliza el ángulo de desviación δ que es calculado a partir del vector dirección del tórax T'_z en el tiempo t, este vector puede ser hallado a partir de un sensor inercial (IMU)(Geiger et al., 2008) que brinda información en tiempo real de la dirección de este vector. Este vector es capaz de medir que tan erguido se mantiene el torso del robot al caminar. Utilizando la ecuación 3.1 se hallará el ángulo de desviación desde $\overrightarrow{T}'_z$, hasta el vector $\overrightarrow{Z} = (0, 0, 1)$ que se muestra en la ecuación 4.38.

$$\delta = angvec\left(\overrightarrow{T}'_z, \overrightarrow{Z}, \overrightarrow{T}'_z \times \overrightarrow{Z}\right) \tag{4.38}$$

Este enfoque de recompensa se basa en que tan erguido esta el torso del robot al momento de caminar, esto favorece el entrenamiento del CM de manera que imita al ser humano, entre más erguido este su torso, mejor equilibrio posee. Esto puede afectar en otros aspectos si el CM que se utiliza es de un ser humano corriendo a una velocidad mayor a 5 km/h, En este caso, el ser humano balancea su cuerpo ligeramente hacia adelante para contrarrestar la resistencia al aire. Por tanto, este método de recompensa esta pensado para una marcha promedio, con una velocidad que entre los 3 y 5 km/h.

Se plantean dos enfoques de función de recompensa para las futuras pruebas:

4.3.1. Estándar

La función de recompensa estándar es dada por el siguiente algoritmo:

$$r_t = \; si \begin{cases} \delta_t \leq u \; y \; t = k & : 1 \\ \delta_t > u & : -1 \\ otro\; caso & : 0 \end{cases} \tag{4.39}$$

donde:

- r_t es la recompensa.
- δ_t es el ángulo de desviación.
- k es el tiempo t final de un CM.
- u es el umbral permitido de inclinación del torso.

Si el torso supera esta inclinación, la recompensa sera -1 y se dará por finalizado el episodio para retornar al tiempo inicial $t = 0$, en caso se logre alcanzar la posición

final $t = k$ y $\delta_t \leq u$ entonces la recompensa es 1 y se regresara al $t = 0$. En cualquier otro caso, la recompensa es 0 y continuará el algoritmo con la siguiente postura mapeada F_{t+1}. Estos valores escogidos corresponden a un estándar de algoritmos de aprendizaje por refuerzo, donde 1 es el premio por lograr el objetivo, -1 el castigo por cometer error y 0 un valor neutral.

4.3.2. Modificada

La función de recompensa modificada está dada por el siguiente algoritmo:

$$r_t = \ si \begin{cases} \delta_t \leq u \ y \ t = k & : \epsilon \\ \delta_t > u & : -1 \\ otro \, caso & : 0 \end{cases} \tag{4.40}$$

donde:

r_t es la recompensa.

δ_t es el ángulo de desviación.

k es el tiempo t final de un CM.

ϵ es una constante de recompensa.

u es el umbral permitido de inclinación del torso.

La diferencia principal con el enfoque estándar es el valor de recompensa que cambio de 1 a un valor ϵ, el cual se recomienda que sea un valor elevado como 100 o 1000. Esto se hace con la finalidad de premiar de forma exagerada una solución satisfactoria dado que son muy escasas. Entre más escasas sean las soluciones correctas, mayor debe ser el valor ϵ.

4.4. Algoritmo de Aprendizaje por Refuerzo

Dado el enfoque de la propuesta, es necesario recurrir a una técnica de aprendizaje por refuerzo, a esta deducción se llegó considerando dos factores. Primero, la necesidad de encontrar un método sencillo de aprendizaje en un espacio de soluciones derivado de la solución anterior en el transcurso del tiempo. Es decir que la solución $(t+1)$ depende de la solución (t) siendo esta solución dependiente en una secuencia de tiempo. Segundo, para lograr buenos resultados se requiere de muchas iteraciones, las cuales demandan tiempo de ejecución alto por cada prueba. Esto hace que sea vital reducir el tiempo de ejecución de las pruebas para una rápida convergencia.

Dado estas características del problema, se buscó un algoritmo de aprendizaje por refuerzo en espacios de estados y acciones continuas que satisfaga los requerimientos del problema, por ello, se propone el uso del algoritmo RBF-Sarsa(λ) (Barrios-Aranibar, 2005). El motivo por el que se utiliza esta técnica fue por su simplicidad y rápida convergencia que puede presentar al tener múltiples dimensiones. Otras técnicas como las redes neuronales profundas (Perervenko, 2016) presentan un convergencia muy lenta con múltiples dimensiones. Sarsa(λ) es un algoritmo que se caracteriza por tomar en cuenta las mínimas variaciones de convergencia dentro del aprendizaje por refuerzo a diferencia del algoritmo Q-leaning (Watkins y Dayan, 1992). La Red Neuronal RBF se caracteriza por su simplicidad y rápida convergencia a diferencia de las Multicapa Perceptron o las redes Neuronales Convolucionales (Pererven ko, 2016) que requieren de mayor tiempo para converger según la cantidad de dimensiones.

Al juntar dos técnicas como SARSA(λ) y la Red Neuronal RBF producen un método de aprendizaje por refuerzo con las características del algoritmo SARSA(λ), que posee la capacidad de tomar en cuenta los estados y acciones pasados para tomar una acción futura, y junto a la Red Neuronal RBF convierte los estados y acciones discretos a continuos almacenando el conocimiento del algoritmo de aprendizaje dentro de las neuronas ocultas de la Red Neuronal RBF.

Existe dos etapas, la etapa de entrenamiento y la etapa de uso. La etapa de entrenamiento se produce cuando se itera con las entradas estados/acción pertenecientes a las neuronas de entrada, y dado los valores, la red escoge una acción futura, la cual es ejecutada y con ayuda de la Función de Recompensa se opinen los valores indicados para que la red neuronal con el conocimiento aprendido hasta el momento pueda escoger una acción futura a partir de todas las perturbaciones echas por la Función de perturbaciones. Este proceso se produce tantas veces como sea necesario hasta que se encuentre una tendencia hacia la convergencia. La etapa de Uso consiste en utilizar la red ya entrenada ingresando el estado actual y generando futuras acciones utilizando la función de perturbación. La red neuronal calificara dichas acciones y selecciona la mejor solución a partir del conocimiento ya aprendido de la Red.

4.5. Medición de similitud

Para medir la similitud que existe entre la solución y el movimiento real del ser humano se propone una ecuación donde se promedia las distancias angulares que existen en cada GDL, normalizado el resultado en el rango mínimo y máximo que cada articulación permita en el modelo, para esto se propone la ecuación 4.41 mostrada a continuación:

$$\delta = 1 - \frac{\sum_{k=1}^{m} \left(\frac{|F_{t_k} - a_{t_k}|}{R_{max_k} - R_{min_k}} \right)}{m} \tag{4.41}$$

donde:

δ es la tasa de similitud.

k es el identificador de cada GDL.

m es la cantidad total de GDL.

F_t es el vector de GDL original mapeado de un ser humano.

a_t es el vector de GDL de la solución tomada.

R_{max} es el vector de rango máximo de cada GDL.

R_{min} es el vector de rango mínimo de cada GDL.

En resumen esta ecuación da como resultado la distancia que existe entre un modelo y otro que tienen la misma arquitectura y los mismos GDL con respecto a una misma secuencia de movimientos. Esta formula propuesta no solo se puede aplicar para arquitecturas bípedas, sino a cualquier arquitectura con n GDL y hiper-redundancia.